ANALYSING THE PERFORMANCE IMPACTS OF LAZY LOADING IN WEB APPLICATIONS

Răzvan-Mihail BÂRA[1]

Costin Anton BOIANGIU[2]

Cătălin TUDOSE[3]

Abstract

This scientific paper explores the effects of lazy loading on the performance of web applications. Lazy loading is a technique employed in web development to defer the loading of non-essential resources until they are effectively needed, thereby optimizing the initial page load times and resource utilization. We'll review a few important sources addressing the problem, we'll examine the most important performance metrics and we'll develop our analysis methodology and experimental setup. This study aims to investigate the impact of lazy loading on various performance metrics, including page load speed, user experience, and overall efficiency of web applications. For comprehensive results, we developed two applications having different architectures (a static website and a website with dynamic content) and we varied the test environment (a high-speed network versus a lower-speed network). The final evaluation would like to conclude about the optimal choice between eager loading and lazy loading in web development and the trade-offs to be made.

Keywords: web development, lazy loading, eager loading, static content, dynamic content, high-speed network, lower-speed network

JEL Classification: O33 Technological Change: Choices and Consequences • Diffusion Processes

1. Introduction

Web applications have become an integral part of our digital lives, serving as dynamic platforms for information dissemination, e-commerce, and interactive user experiences. As user expectations continue to rise, so does the demand for web applications that deliver high

[1] Student, POLITEHNICA National University for Science and Technology of Bucharest, Romania, razvan_mihail.bara@stud.etti.upb.ro

[2] PhD, Professor, POLITEHNICA National University for Science and Technology of Bucharest, Romania, costin.boiangiu@upb.ro

[3] PhD, Lecturer, POLITEHNICA National University for Science and Technology of Bucharest and Luxoft Romania, Romania, catalin.tudose@gmail.com

performance. One critical aspect of optimizing web application performance is the loading time of resources during the initial page load. Users often associate a sluggish or delayed loading experience with poor application performance, which can lead to frustration and decreased user engagement, having as ultimate consequence people no longer accessing the web application. [1]

In response to this challenge, web developers employ various techniques to enhance loading efficiency. Lazy loading has emerged as a prominent strategy, allowing developers to defer the loading of non-essential resources until they are required for user interaction. This deferred loading aims to streamline the initial page load, reduce bandwidth consumption, and improve overall user experience. The concept of lazy loading has gathered significant attention as a means to strike a balance between feature-rich web applications and optimal performance. [2]

The primary objective of this study is to delve into the effects of lazy loading on the performance of web applications. By conducting a systematic analysis, we aim to uncover insights into how lazy loading influences key performance metrics, including page load speed, network usage, and user experience. Through this investigation, we seek to provide web developers with a deeper understanding of the trade-offs associated with lazy loading, enabling them to make informed decisions in the pursuit of creating high-performance web applications.

To achieve this goal, we will employ a rigorous experimental methodology, using real-world web applications and several test scenarios. The study will not only assess the quantitative impact on performance metrics but also explore the qualitative aspects of user interactions and satisfaction. By addressing these aspects, we aim to contribute valuable knowledge that can guide best practices for implementing lazy loading in web development.

In the subsequent sections, we will review existing literature on lazy loading, detail our experimental setup and methodology, present and analyze results, and conclude with recommendations for developers and plans for future research. Through this exploration, we aspire to contribute to the ongoing dialogue on web application optimization and advance the understanding of the practical implications of lazy loading in real-world scenarios.

2. Literature Review

The literature review aims to provide a comprehensive overview of existing research on lazy loading in web applications. It examines key studies and findings related to the impact of lazy loading on web application performance, user experience, and overall efficiency.

2.1 Lazy Loading in Web Development

Lazy loading has gained prominence as a technique for optimizing web application performance by deferring the loading of non-essential resources until they are needed. [3] According to Simon Fray, lazy loading minimizes the initial page load times and enhances user experience by prioritizing critical content. Additionally, lazy loading reduces server loads and conserves bandwidth, contributing to a more sustainable and scalable web infrastructure. [4]

2.2 Performance Metrics in Lazy Loading Studies

Jamie Juviler highlights the importance of considering multiple performance metrics when evaluating the effects of lazy loading. This study analyzes the impact on page load speed, resource utilization, and user engagement. [2]

2.3 User Experience Implications

User experience is a crucial aspect of web application development. The study by Jamie Juviler [2] investigates the correlation between lazy loading and user satisfaction, emphasizing the positive impact on perceived performance. On the opposite side, Felix Arntz and Rick Viscomi [3] suggest potential drawbacks, such as delayed rendering of images during user interaction, raising concerns about the trade-offs associated with lazy loading.

3. Methodology

3.1 Experimental Setup

To assess the effects of lazy loading on web application performance, we conducted experiments on two distinct applications: a static website and a dynamic website representing an online library. Also, we varied the environmental conditions between a high-speed network and a lower-speed network.

3.1.1 Static Website

The first application is a static website designed to showcase content without dynamic interactions. This serves as a baseline scenario to evaluate the impact of lazy loading on straightforward, content-focused web pages. The website consists of one single page, including 5 sections, each section containing a slider of 10 high-quality images.

3.1.2 Full-Stack Application for an Online Library

The second application represents a comprehensive full-stack online library, featuring dynamic content, user authentication, and real-time interactions. This complex application allows us to explore the scalability and adaptability of lazy loading in a more interactive and data-intensive environment.

3.2 Performance Metrics

Our methodology involves measuring various performance metrics to comprehensively evaluate the impact of lazy loading on both applications. The key metrics include:

3.2.1 Initial Load Time

We will measure the time it takes for the initial load page to completely load, starting from the initiation of the request to the rendering of all visible content. This metric serves as a fundamental indicator of the overall responsiveness of the applications.

3.2.2 Bandwidth Usage

To assess the efficiency of lazy loading in conserving bandwidth, we will monitor the amount of data transferred between the server and the client during the page loading process. This measurement is crucial for understanding the potential benefits of lazy loading, particularly in scenarios with limited network resources.

3.2.3 User Experience Metrics

We use the Chrome DevTools and the Lighthouse extension to gather user experience metrics, including First Contentful Paint (FCP), Time to Interactive (TTI), and Cumulative Layout Shift (CLS). These metrics provide insights into the perceived performance and visual stability of web applications.

First Contentful Paint (FCP) is an important metric, centered on the user and measuring the perceived load speed. It indicates the first moment during the loading of the page when the user sees something being displayed. A quick FCP gives the user the impression that things are moving on.

Time to Interactive (TTI) indicates the earliest time after First Contentful Paint (FCP) when the page is ready to interact with the user. This performance metric helps identify situations where a page erroneously looks interactive by measuring a page's load responsiveness.

Cumulative Layout Shift (CLS) examines the unexpected movement of elements in the viewport as the page loads. This is a measure of the visual stability of the content of the page, as unforeseen layout shifts can displease the user.

3.3 Test Cases

For each application, we designed a series of test cases to simulate different user interactions and scenarios. These include:

3.3.1 Navigation Scenarios

- Initial page load
- Navigation between pages
- Loading additional content dynamically

3.3.2 User Interaction Scenarios

- Scrolling through content
- Interacting with dynamic elements (e.g., forms, modals)

3.4 Tools and Technologies

The experiments were conducted using Google Chrome's DevTools and the Lighthouse extension. The DevTools provide in-depth insights into the network activity, rendering performance, and user interactions, while the Lighthouse extension automates the collection of performance metrics and provides a standardized evaluation of web page performance.

3.5 Test Environments

All experiments were conducted in controlled environments to minimize external factors affecting the measurements. The test environments included standardized hardware specifications and network configurations to ensure consistency across experiments.

We will define two environments for testing, to examine the effects of lazy loading in both cases:

- A high-speed network, referenced as HSN
- A lower speed network, referenced as LSN

4. Results and Analysis

4.1 Static Website

This application consists of a single HTML page to showcase different landforms. It has 5 distinct sections, each of them displaying 10 pictures of a landform. We compare the results

of using lazy-loading performance versus eager-loading performance in the following subsections.

4.1.1 Initial Load Time

For all 50 pictures to load, using the eager loading strategy the initial page load time is 5.5 seconds. With the 5G network (a high-speed network), using lazy loading, the initial load time is only 0.5 seconds. The load time decreases drastically as more and more images are lazy-loaded, as described in Figures 1 and 3.

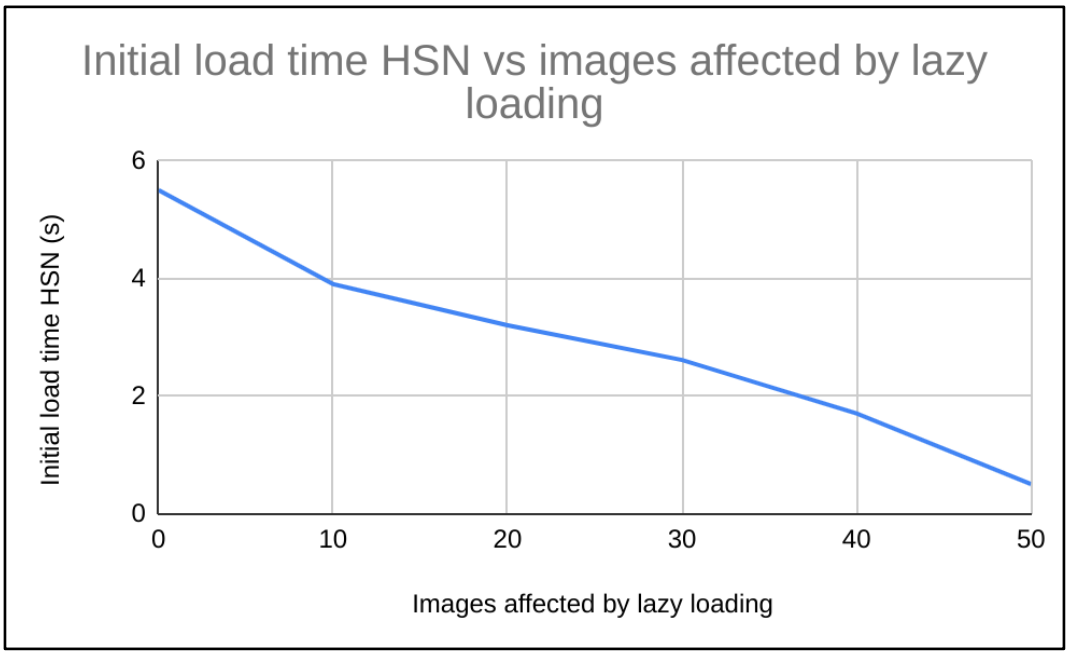

Figure 1. Graph describing the decrease of the initial load time of the HSN

This measurement is heavily impacted by the type of network in use. For example, when the user is connected to a 3G network (a lower-speed network), the total time for eager loading is 1.7 minutes while for lazy loading is only 6.9 seconds.

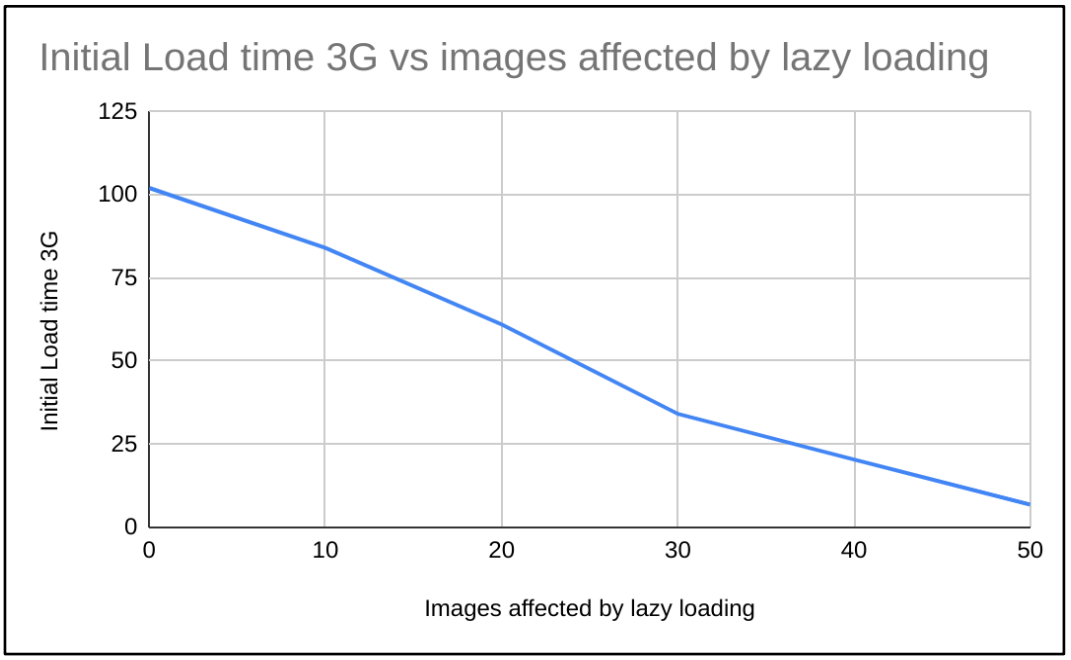

Figure 2. Graph describing the decrease of initial load time for the LSN

In the context of network performance, lazy loading proves to be especially beneficial for a 3G network compared to a 5G network. In a 3G environment, where bandwidth is more limited, lazy loading contributes significantly to a smoother and more responsive user experience. By deferring the loading of images, until they are needed, lazy loading conserves bandwidth and reduces the strain on the network.

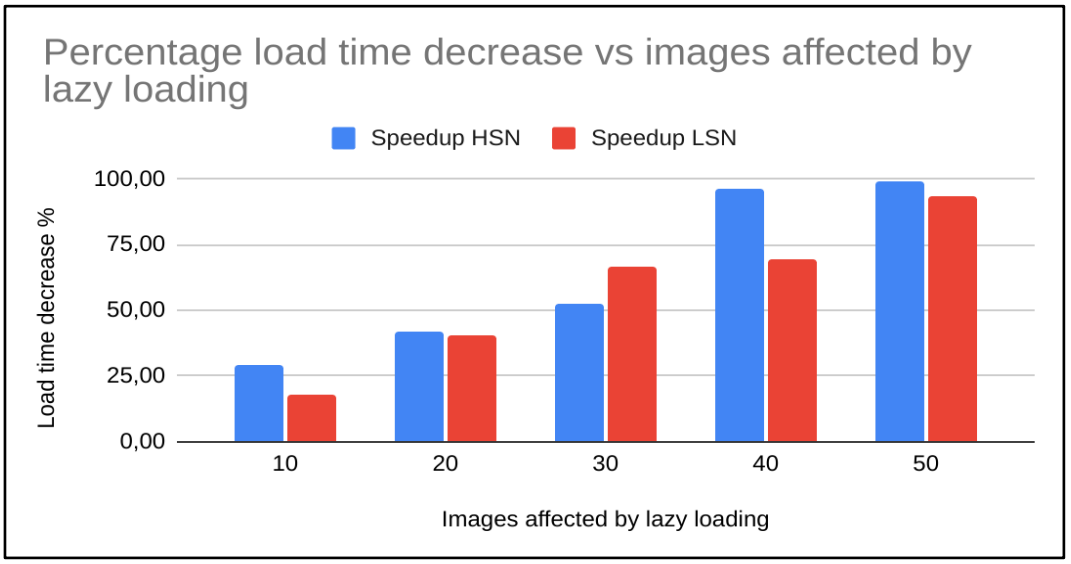

Figure 3. Speedup of HSN and LSN

4.1.2 Network Utilisation

Also, as more and more images are lazy-loaded, the bandwidth usage of the web page decreases. The page transfers about 18 MB of image content through the network. The same amount is transferred when using the lazy loading strategy, but only if the user explores the whole page. Otherwise, the amount of transferred data depends on the percentage of the images requested by the user when navigating to a certain part of the page.

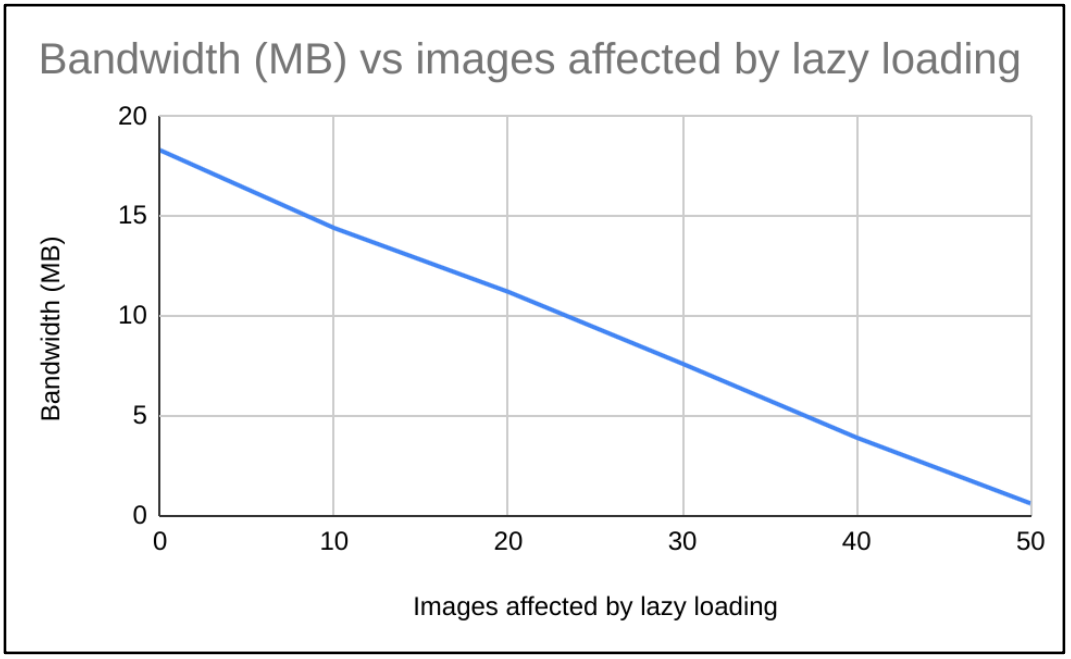

Figure 4. Decrease of initial bandwidth usage

This table displays the data that has been considered for this study:

Lazily loaded images	Bandwidth (MB)	Initial load time 5G (s)	Speedup 5G (%)	Initial load time 3G (s)	Speedup 3G (%)
0	18.3	5.5	0	102	0
10	14.4	3.9	29.09	84	17.65
20	11.2	3.2	41.82	61	40.20

Lazily loaded images	Bandwidth (MB)	Initial load time 5G (s)	Speedup 5G (%)	Initial load time 3G (s)	Speedup 3G (%)
30	7.6	2.61	52.55	34.1	66.57
40	3.9	1.7	95.93	20.37	69.40
50	0.6	0.5	99.05	6.8	93.33

Table 1 – Measurements of the static website

4.1.3 User Experience

While the slow initial page load time of eager loading would make a user leave the website due to the impression that the application isn't working properly, the navigation experience would prove itself to be a pleasant one. All the content is already loaded onto the page, so scrolling and exploring through the different sliders of landforms presents no issue.

On the other hand, for lazy loading, the initial page load time is fast, but the user might encounter some issues while exploring the page. If the network is slow, the user might end up leaving because of the on-demand loading time of a picture. If the network speed is fast enough, the experience of exploring the sliders would be like the one of eager loading. As a result, users can access and interact with the core content of the webpage faster, without being hindered by the need to wait for all images to load simultaneously. This not only enhances the overall performance of the website but also ensures that users can quickly engage with the essential information, creating a seamless and more satisfying browsing experience.

4.1.4 Lighthouse extension reports

For eager loading, both the first contentful paint and the largest contentful paint took about 1.3 seconds to render. For the other strategy, the first contentful paint took 0.5 seconds and the largest contentful paint took 0.6 seconds. The first metric refers to how fast a user can see an element on the page, while the second metric refers to how fast a user can see the largest element.

Because of its static web page nature, the SEO score is the same for both loading strategies.

4.2 Full-Stack Application for an Online Library

This application represents the graphical interface for an online library, which can be used by users to browse the available book collection and place book orders. The books, authors, categories of books, and users can also be managed from the admin panel of the application.

The development of this project was done using Typescript and the Angular framework. It contains two main modules, a public one and a private one. The public one is available to all users, even if they're unregistered, while the private one is dedicated to the managerial needs of the library. There is a total number of 18 Angular modules, containing bundles of HTML, CSS, and Typescript components. As a default setting, the Angular framework uses eager loading.

One of the most obvious uses of lazy loading in the context of this application is to separate the loading of user and admin modules.

4.2.1 Initial Load Time

Breaking down a large codebase into smaller, focused modules, promotes a more manageable and understandable structure, making it easier for developers to collaborate and maintain the code over time. Additionally, the modular approach facilitates code reuse, as individual modules can be utilized across different parts of a project or even in other projects, reducing redundancy and promoting a more efficient development process. Furthermore, by loading only the necessary modules when needed, there is a performance benefit, as it minimizes the initial load time of a web page.

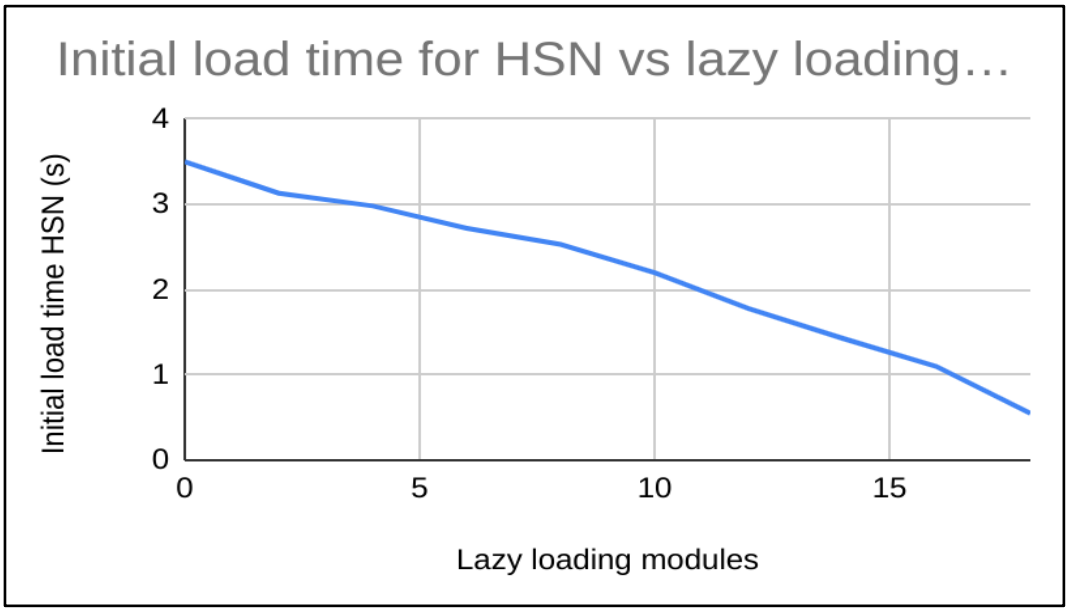

Figure 5. Initial load time HSN vs lazy loading modules

Modules are lazily loaded from the bottom up, starting from the smaller ones. As expected, when lazy loading the admin module, the load time decreases the most, compared to the other measurements, 0.6 seconds.

The slower nature of the 3G network is easily observed here along with the benefits of lazy loading the modules, as the load time decreased from 132 seconds to 40 seconds.

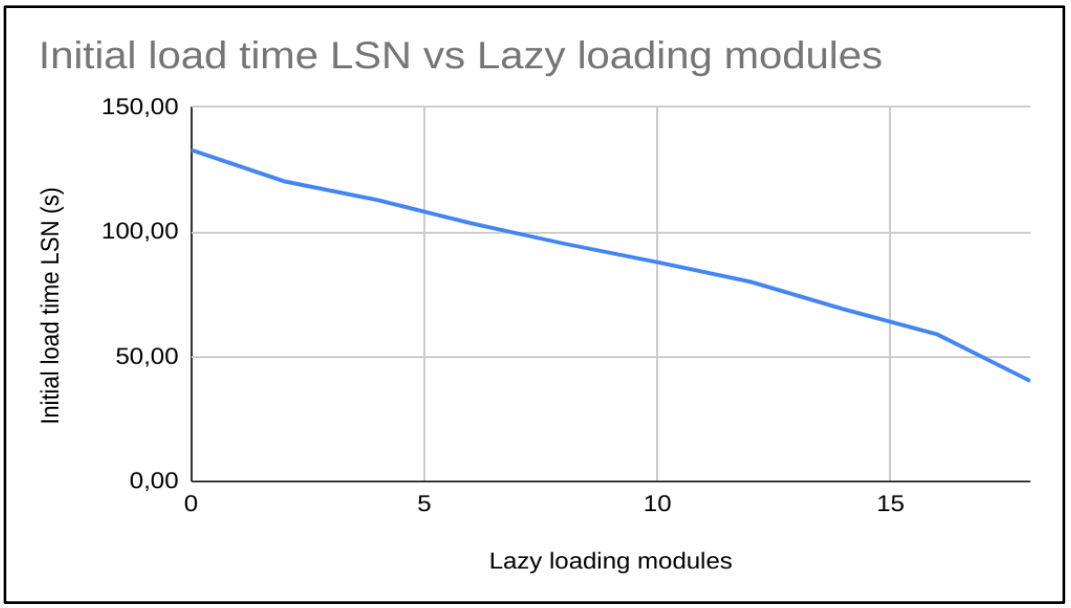

Figure 6. Initial load time for LSN vs lazy loading modules

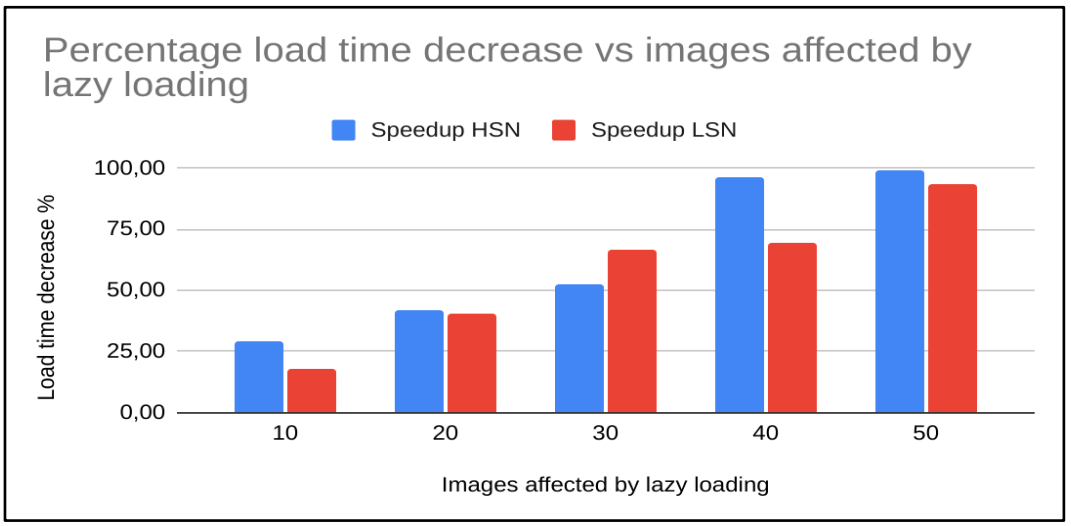

Figure 7. Speedup of HSN and LSN

On the other hand, the constrained bandwidth results in slower data download speeds, leading to delayed loading of web pages, especially those with large or numerous assets such as images, scripts, and stylesheets. So, as a consequence of the slower 3G network, the speedup effect is not as effective as the previous network.

4.2.2 Network Utilization

As anticipated, as the number of lazy-loaded modules rises, so does the initial bandwidth demand. While using an API to retrieve data from a database, this application is not as representative as the previous static website example. Since lazy loading only delays the loading of Angular components that constitute the user interface, the content is therefore dynamic and the amount of data transferred while visiting the same page may vary at different times.

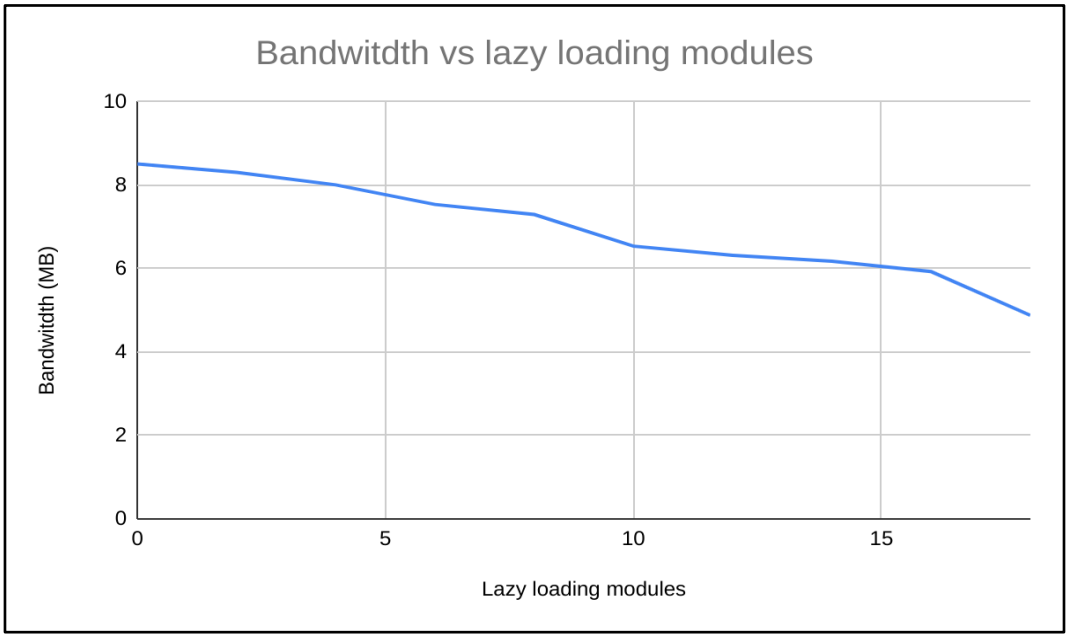

Figure 8. Bandwidth usage

4.2.3 User Experience

The decision to load Angular modules eagerly or slowly has a big impact on how a web application feels to its users. Because only the relevant modules are loaded when needed, lazy loading has the advantage of faster initial page loads, improving user responsiveness. As a result, the connection runs more smoothly overall and bandwidth is used more effectively. However, because the necessary modules are loaded dynamically, users may experience a brief lag when they browse to a new part. Positively, lazy loading reduces the

initial resource footprint and improves scalability, which is especially useful for larger applications.

On the other hand, eager loading guarantees that every module loads during the first page load, giving users instant access to all of the application's features. Although this method removes waiting times when navigating between sections, it frequently results in lengthier initial load times, which may affect how quickly the application is perceived, especially for slower network connections. For smaller applications, when the trade-off between initial load time and instant access to functionality is more acceptable, eager loading is beneficial.

To maximize the overall user experience, the two solutions should be carefully assessed depending on the unique features and needs of the Angular application.

4.2.4 Lighthouse extension reports

Single Page Application (SPA) frameworks like the ones developed with Angular can face SEO challenges due to their dynamic content-loading approach. SPAs typically load a minimal HTML shell initially and fetch the content using JavaScript, raising difficulties for search engine crawlers to effectively index content and interpret unique meta information for each page. Traditional websites, which render different HTML files for each page, benefit from clearer link structures and specific SEO elements. While some search engine crawlers have improved their JavaScript execution capabilities, challenges persist, leading to potential delays in indexing and ranking SPAs. Developers often employ strategies like server-side rendering or prerendering to generate static HTML snapshots, making SPAs more SEO-friendly by providing better crawlable and indexable content.

Therefore, after using the Lighthouse extension on different pages of the application, with and without lazy loading the **SEO score** is fairly low compared to a traditional website that renders different HTML files to the user. So, both approaches offer similar results.

5. Conclusions

To sum up, deciding between eager loading and lazy loading in web development requires careful evaluation of trade-offs. Especially in larger applications, lazy loading helps optimize initial page load times, save bandwidth, and improve overall performance with its on-demand module loading approach. Although there may be a small delay when switching between sections, this approach encourages a more responsive user experience.

However, eager loading guarantees that all of an application's features are available right away during the initial load. This might be useful for smaller applications, but it may result in lengthier initial load times.

Deciding about applying lazy loading is a contextual matter. It may be preceded by several experiments, varying the important parameters related to the application (content size,

content type, number of loaded modules) or the environment (network speed and configuration).

References

[1] A. TURCOTTE, S. GOKHALE and F. TIP, *Increasing the Responsiveness of Web Applications by Introducing Lazy Loading*, 2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE), Luxembourg, Luxembourg, 2023, pp. 459-470, doi: 10.1109/ASE56229.2023.00192.

[2] J. JUVILER, *Lazy Loading: How It Decreases Load Time and Increases Engagement* https://blog.hubspot.com/website/lazy-loading-eager-loading

[3] X. S. WANG, A. BALASUBRAMANIAN, *Demystifying Page Load Performance with WProf*, Proceedings of the 10th USENIX conference on Networked Systems Design and Implementation, April 2013, Pages 473–486

[4] S. FRAY, *Lazy Loading VS Eager Loading || What Impact Do They Have?*, https://simon-frey.com/blog/lazy-loading-vs-eager-loading

[5] F. ARNTZ, R. VISCOMI, *The performance effects of too much lazy loading*, https://web.dev/articles/lcp-lazy-loading#causal_performance

Bibliography

F. ARNTZ, R. VISCOMI, *The performance effects of too much lazy loading*, https://web.dev/articles/lcp-lazy-loading#causal_performance

A. BLANKSTEIN, S. SEN, M.J. FREEDMAN, *Hyperbolic Caching: Flexible Caching for Web Applications*, 2017 USENIX ANNUAL TECHNICAL CONFERENCE (USENIX ATC '17), Page 499-511

S. FRAY, *Lazy Loading VS Eager Loading || What Impact Do They Have?*, https://simon-frey.com/blog/lazy-loading-vs-eager-loading

F. AL-HAWARI, *Software design patterns for data management features in web-based information systems*, Journal of King Saus University-Computer and Information Sciences, Volume34 Issue10 Page 10028-10043, Part B, doi: 10.1016/j.jksuci.2022.10.003

J. JUVILER, *Lazy Loading: How It Decreases Load Time and Increases Engagement* https://blog.hubspot.com/website/lazy-loading-eager-loading

E. MJELDE, A. OPDAHL, *Load-Time Reduction Techniques for Device-Agnostic Web Sites, Journal of Web Engineering*, Volume 16, Issue3-4, Page 311-346, June 1, 2017

M.D. SALAS-ZARATE, G. ALOR-HERNANDEZ, R. VALENCIA-GARCIA, L. RODRIGUEZ-MAZAHUA, A. RODRIGUEZ-GONZALEZ, J.L.L. CUADRAD, *Analyzing best practices on Web development frameworks: The lift approach*, Science of Computer Programming, Volume 102, Page 1-19, doi: 10.1016/j.scico.2014.12.004

N. TUAYCHAROEN, V. PRODPRAN, B. SRITHONG - *ClassSchedule: a Web-based Application for School Class Scheduling with Real-time Lazy Loading*, Proceedings of 2018 5th International Conference on Business and Industrial Research (ICBIR), Page 210-214

A. TURCOTTE, S. GOKHALE and F. TIP, *Increasing the Responsiveness of Web Applications by Introducing Lazy Loading*, 2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE), Luxembourg, Luxembourg, 2023, pp. 459-470, doi: 10.1109/ASE56229.2023.00192.

X. S. WANG, A. BALASUBRAMANIAN, *Demystifying Page Load Performance with WProf*, Proceedings of the 10th USENIX conference on Networked Systems Design and Implementation, April 2013, Pages 473–486

A.K.I. YASARI, A.D. ABBAS, H.A. ATEE, L.A. LATIFF, R.A. DZIYAUDDIN, D.A. HAMMOOD, *Lazy Multi-Level Dynamic Traffic Load Balancing Protocol for Data Center (LMDTLB)*, Journal of Engineering Science and Technology, Volume 16, Issue 3, Page 2439-2453, June 2021