BULLETIN of Cherkasy State Technological University

Journal homepage: https://bulletin-chstu.com.ua/en

Vol. 30 No. 2. 2025

UDC 681.3.07
DOI: 10.62660/bcstu/2.2025.88

Optimising web interface performance using Amdahl's law

Oleh Prus*
Postgraduated Student
Vinnytsia National Technical University
21021, 95 Khmelnytske Shose Str., Vinnytsia, Ukraine
https://orcid.org/0009-0007-2514-9871
Volodymyr Maidaniuk
PhD in Technical Sciences, Associate Professor
Vinnytsia National Technical University
21021, 95 Khmelnytske Shose Str., Vinnytsia, Ukraine
https://orcid.org/0000-0002-3345-2578

Abstract. The article discussed the construction and application of generalised mathematical models based on Amdahl's law to determine the maximum possible acceleration of web interfaces, taking into account their key features. An extension of the classical approach was proposed by including asynchronous processes, multi-level caching mechanisms, and dynamic resource loading methods in the model, which allows for a more accurate assessment of the cumulative impact of various optimisations on performance. In particular, the feasibility of taking into account asynchronous data exchange was justified, which allows processing requests in parallel and avoiding blockages in the process of updating content. A formula has been developed that takes into account the effectiveness of client and server caches and provides a quantitative assessment of the reduction in response time when reusing already loaded data. Particular attention was focused on step-by-step content retrieval techniques, where the initial page load was minimised by deferring the addition of individual scripts, images or styles, which speeds up the initial display of important content and makes the interface more responsive to user actions. In addition, the impact of a comprehensive combination of optimisation strategies on web interface performance was considered, and a corresponding generalised model was proposed, which uses an interdependence coefficient to determine the extent to which one optimisation enhances or, conversely, negates the effect of another. This makes it possible to predict the total performance gain and compare the cost of implementing several solutions with the potential time savings. The proposed formalised approach can serve as a basis for creating automated tools for evaluating web interface performance, integrated into the development process. Testing the model in three practical scenarios – partial rendering with API caching, JavaScript minification with a content delivery network (CDN), and code splitting with server-side caching – yielded performance gains of 1.87 ×, 1.55 ×, and 1.64 ×, respectively, which was fully consistent with theoretical predictions. The data obtained confirmed the ability of the *R* interdependence coefficient to accurately reflect the synergy or overlap of optimisation effects and makes the model suitable for pre-selecting the most effective acceleration strategies at the CI/CD audit stage

Keywords: mathematical models; asynchronous execution; data caching; dynamic loading; combined strategies; performance prediction

Article's History: Received: 03.04.2025; Revised: 24.04.2025; Accepted: 16.06.2025.

Suggested Citation:
Prus, O., & Maidaniuk, V. (2025). Optimising web interface performance using amdahl's law. *Bulletin of Cherkasy State Technological University*, 30(2), 88-96. doi: 10.62660/bcstu/2.2025.88.

*Corresponding author

INTRODUCTION

Optimising the performance of web interfaces is one of the priorities of modern web development, as speed and ease of interaction with the user environment directly affect visitor retention and the business performance of digital platforms. In the design of high-load web systems, the integration of caching mechanisms, asynchronous requests, and dynamic resource loading has become a key area of development, but there is still no universal methodology for the formalised evaluation of the effectiveness of these optimisations. The relevance of the problem was confirmed by the growing demands for stable service operation amid rapid increases in data volumes and user numbers.

Researchers such as M.H. Kundos *et al*. (2024) were paying considerable attention to the problem of reducing page load times and minimising rendering delays, which is due to the rapid growth in the complexity of web applications. Some researchers focused on the empirical analysis of individual acceleration techniques, such as partial rendering or the use of distributed network infrastructure. CDN, however, lacks generalised mathematical models that would allow accurate prediction of the cumulative impact of various optimisations at the planning stage. R. Abounacer *et al*. (2023) believed that in this context, Amdahl's law, which is traditionally used to estimate the maximum possible acceleration of a system after optimising a specific part of it, offers a promising theoretical basis. However, the classical formulation does not take into account a number of factors characteristic of web applications, in particular asynchronous execution, multi-level caching, and load variability caused by user behaviour dynamics. In the field of web technologies, this approach is used to analyse the impact of various performance improvement methods, including partial rendering, data caching, and network request optimisation, as can be seen in the work of A. Nair *et al*. (2024). This study reviewed key research that applies Amdahl's law in the context of web development, evaluating its application for determining the effectiveness of optimisations in multi-project environments.

The article by L. Mitton (2023) examined Amdahl's law formula and its application for predicting the potential acceleration of tasks when improving system resources. The author emphasised that even when additional processors are added, the maximum acceleration is limited by the part of the task that cannot be parallelised. The limitations of Amdahl's law were also discussed, in particular its unsuitability for cases with variable workloads. The article provided a general understanding of Amdahl's law, but did not consider its application in the context of web interfaces. Further research may focus on adapting this law to evaluate optimisations in web applications, taking into account the specifics of their architecture and user interaction. In addition, the extension of the classic Amdahl formula to multiple resources, proposed by C. Poolla & R. Saxena (2022), confirmed the accuracy of the multidimensional regression model (≈ 95% cross-validation) for predicting total acceleration when scaling multiple subsystems simultaneously. This approach formed the basis of the interdependence coefficient R and allowed for a formal assessment of the synergy or conflict between optimisation strategies.

At the same time, some researchers are focusing on specific aspects of improving the performance of modern web solutions that could be integrated into the extended formulation of Amdahl's law. For example, I. Malavolta *et al*. (2020) and T. Majchrzak *et al*. (2018) studied the impact of caching on the efficiency and overall performance of progressive web applications (PWA), emphasising the advisability of considering cache mechanisms as an important component of optimisation. At the low-level optimisation level, J. Ren *et al*. (2024) proposed JSTUNER, an ML-oriented auto-tuning system that provides an average acceleration of JS code execution by ×1.62 on mobile devices by automatically selecting the optimal sequences of Chrome V8 compiler flags. This confirmed that intelligent approaches can complement classic manually tuned optimisations. A comprehensive review by J. Vepsäläinen *et al*. (2024) showed that the average mobile page size has increased sixfold over the past decade, and therefore the need for systematic optimisation methods is only growing. The authors highlighted code splitting, caching, and CDN as the most effective areas, which correlates with the topic of this work. Regarding single-page applications, M. Kothapalli (2022) empirically showed that combining code splitting and lazy loading reduces the first rendering time of SPA by 40%, and additional caching ensures stable offline operation.

Thus, modern works from 2020-2024 cover the entire spectrum – from ML-tuning of compilers and SPA portfolios to CDN edge platforms and analytical extensions of Amdahl's law – which justifies the relevance of the generalised model developed in the article for predicting the combined performance gain of web interfaces. The generalisation of the theoretical provisions of Amdahl's law with modern research developments in the field of caching, asynchronous operations, dynamic loading and scalable testing makes it possible to form a more comprehensive model for optimising web interfaces. This confirms the need for further theoretical and practical research aimed at improving the tools for determining the maximum possible acceleration of web applications of different architectures and scales. Therefore, despite the fundamental nature of Amdahl's law, its application in the context of web interface optimisation requires further research. The development of mathematical models and practical recommendations for the evaluation of effectiveness of optimisations in multi-project web environments will increase application productivity and improve user experience, as noted in the work of O. Prus *et al*. (2024).

In view of the above, the aim of this study was to test Amdahl's law adapted to the specifics of web development, in particular to build an extended mathematical model for evaluating interface performance, taking into account asynchronous processes, dynamic loading, and caching mechanisms. To achieve this goal, the following tasks were defined: to analyse modern approaches to optimising the performance of web applications and their empirical effectiveness; to adapt the existing formulation of Amdahl's law, taking into account the specifics of asynchronous and cached operations; to develop a generalised model that will allow evaluating the impact of several optimisations simultaneously and determining the interdependence between them; to verify the validity of theoretical propositions on real examples, conduct numerical experiments and compare the results obtained with data published in recent scientific publications.

The scientific novelty lies in the fact that, for the first time, a formalised approach to measuring the combined effect of several parallel optimisations in a web environment has been proposed, filling the gap between empirical testing methods and the theoretical acceleration model. The use of the adapted Amdahl's law will allow developers and operations engineers to make more informed decisions about choosing performance improvement strategies, taking into account the potential limitations and interactions of different approaches.

MATERIALS AND METHODS

In the classic formulation, Amdahl's law estimates the maximum possible acceleration of a system after optimising its part, which is determined by the equation:

$$S = \frac{1}{(1-P)+\frac{P}{N}}, \quad (1)$$

where S – total system acceleration; P – the proportion of the program to be optimised; N – the acceleration coefficient of the optimised part (GeeksForGeeks, 2023). However, the classic formulation does not take into account the specifics of web interfaces, where performance depends not only on serial and parallel execution, but also on asynchronous processes, caching, and dynamic resource loading. For the correct application of Amdahl's law in web development, it is necessary to expand its model by including additional factors (Junghans *et al.*, 2017).

In the traditional case, acceleration is only possible through parallel computing. However, in web interfaces, a significant proportion of processes are asynchronous requests to the server (e.g., AJAX, Fetch API, WebSockets) and deferred loading of components (e.g., lazy loading) (Ahmed *et al.*, 2017, Oh *et al.*, 2025). When using asynchronous acceleration mechanisms, a web application can be described as:

$$S_{async} = \frac{1}{(1-P_{sync}-P_{async})+\frac{P_{sync}}{N}+\frac{P_{async}}{A}}, \quad (2)$$

where S_{async} – total system acceleration, taking into account asynchronous acceleration mechanisms; P_{sync} – proportion of serial code (not subject to optimisation); P_{async} – proportion of code executed asynchronously; N – acceleration coefficient of the optimised part; A – acceleration coefficient due to asynchronous execution (determined by the efficiency of parallel request processing).

This modification made it possible to more accurately predict the performance of the web interface when using asynchronous API requests, distributed data processing, and dynamic content updates. Caching is one of the main mechanisms for optimising the performance of web applications. Its use has made it possible to reduce the number of requests to the server, reducing page loading and rendering times. According to G. Araújo *et al.* (2024), caching is analysed at three levels: client caching (browser, Service Workers), server caching (Memcached, Redis), and CDN caching (Cloudflare, Akamai).

The impact of caching can be assessed by introducing a cache efficiency coefficient C, which determines what proportion of requests reuse cached data. Taking caching into account, Amdahl's law has been modified as follows:

$$S_{cache} = \frac{1}{(1-P_{cache}-P_{non-cache})+\frac{P_{cache}}{C}+\frac{P_{non-cache}}{N}}, \quad (3)$$

where S_{cache} – total system acceleration taking into account caching; P_{cache} – the proportion of requests that can be processed by the cache; $P_{non\text{-}cache}$ – the proportion of requests that need to be processed by the server; N – the acceleration coefficient of the optimised part; C – the cache efficiency coefficient, which shows how much faster cached requests are processed. According to a study by M. Dyvak & O. Kindzerskyi (2024), the inclusion of caching in the model made it possible to assess the impact of different caching strategies on web interface performance and to draw conclusions about the feasibility of implementing cache mechanisms in specific scenarios (2024).

As noted by R. Felani *et al.* (2020), modern web applications use dynamic content loading, which reduces the initial page load time. These methods include: lazy loading of images, scripts, and styles; code splitting, which allows JavaScript modules to be loaded only when needed; and progressive rendering, which speeds up the display of the first page content. The impact of these mechanisms on performance can be represented as:

$$S_{dynamic} = \\ = \frac{1}{(1-P_{dynamic}-P_{sync}-P_{async})+\frac{P_{sync}}{N}+\frac{P_{async}}{A}+\frac{P_{dynamic}}{D}}, \quad (4)$$

where $S_{dynamic}$ – total system acceleration, taking into account dynamic content loading; $P_{dynamic}$ – the proportion of resources that are loaded dynamically; P_{sync} – the proportion of serial code (not subject to optimisation); P_{async} – the proportion of code executed asynchronously; N – the acceleration coefficient of the optimised part;

A – acceleration coefficient due to asynchronous execution (determined by the efficiency of parallel request processing); D – acceleration coefficient due to dynamic loading (determined by the efficiency of distributed loading resources) (Ma, 2024).

Practical significance: including dynamic loading in the Amdahl's law model helps figure out how effectively lazy loading, code splitting, and progressive rendering techniques are used in a specific web app. Taking all the factors into account, the generalised Amdahl's law model (5) for web apps looks like this:

$$S_{web} = \frac{1}{(1-P_{sync}-P_{async}-P_{cache}-P_{dynamic})+\frac{P_{sync}}{N}+\frac{P_{async}}{A}+\frac{P_{cache}}{C}+\frac{P_{dynamic}}{D}}. \quad (5)$$

This model made it possible to quantitatively assess the impact of asynchronous execution, caching, and dynamic resource loading on the overall performance of a web application. The extended mathematical model of Amdahl's law takes into account key features of web interfaces that affect performance, in particular: asynchronous execution, caching mechanisms, and dynamic resource loading. The proposed formula made it possible to quantitatively assess the effectiveness of various optimisation methods and predict their impact on the performance of web applications.

The next stage of the research was to model combined optimisations, which made it possible to determine the overall effect of simultaneously applying several optimisation methods in web applications. In real web applications, it is rare to use only one optimisation method. More often, combinations of optimisations are used, for example: partial rendering combined with API request caching (reducing delays when updating content); combining JavaScript files and using CDN (reducing network load); browser-level caching and server-side caching (minimising the processing of repeat requests).

In its classic formulation, Amdahl's law evaluates the effectiveness of only one optimisation at a time. However, when several acceleration methods are applied simultaneously, their effect may not be simply cumulative, but interdependent, which makes it difficult to predict the effectiveness of combined optimisations. Thus, the main task in solving this problem was to develop a mathematical model that allows evaluating the overall effect of the simultaneous application of several optimisation methods. The next step was to formalise combined optimisations within the Amdahl's law. Two optimisations acting on different parts of the system are considered as follows: optimisation accelerates the share with a coefficient ; optimisation accelerates the share with a coefficient . If these optimisations are independent of each other, their combined effect can be estimated by the formula:

$$S_{combined} = \frac{1}{(1-P_A-P_B)+\frac{P_A}{N_A}+\frac{P_B}{N_B}}. \quad (6)$$

However, in real web applications, optimisations often overlap. For example, if both API request caching and partial rendering are used, the overall effect may be lower than expected due to the overhead of synchronisation between the two mechanisms. To account for the interdependence between optimisations, an interdependence coefficient was introduced, which determines how strongly one optimisation affects the effectiveness of another. Then the general formula will look like this:

$$S_{combined} = \frac{1}{(1-P_A-P_B+RP_AP_B)+\frac{P_A}{N_A}+\frac{P_B}{N_B}}, \quad (7)$$

where R – interdependence coefficient (values from -1 to 1). If $R > 0$, optimisations reduce each other's effectiveness. If $R < 0$, optimisations complement each other and provide an additional effect. If $R = 0$, optimisations act independently. This model allows for performance evaluation to be adjusted depending on whether the optimisations interact with each other or work autonomously. The next step in the study was to create a generalised model for several optimisations. If we expand the model for n optimisations, the following was obtained:

$$S_{multi} = \frac{1}{(1-\sum_{i=1}^{n}P_i+\sum_{i=1}^{n}\sum_{i=i+1}^{n}R_{ij}P_iP_j)+\sum_{i=1}^{n}\frac{P_i}{N_i}}, \quad (8)$$

where P_i – the part of the process that is accelerated by the i-th optimisation; N_i – acceleration coefficient for optimisation i; R_{ij} – coefficient of interdependence between optimisations i and j. The resulting formula made it possible to evaluate the effectiveness of any number of optimisations, taking into account their mutual influence.

During testing of the created mathematical model, it became necessary to demonstrate how the combined use of several optimisation methods affects the overall performance of a web application. For this purpose, two examples were selected, which present different scenarios of acceleration combinations (partial rendering with caching and JavaScript minification in combination with CDN). Such a comparison makes it possible to trace the conditions under which optimisations reinforce each other, and under which their effects may overlap in a certain way or even reduce the overall result. In this study, the parameters N and P were evaluated using web application performance profiling tools, including Chrome DevTools, Lighthouse, and WebPageTest (Potdar *et al.*, 2020). The P code share was defined as the ratio of the execution time of a specific processing stage (e.g., rendering or waiting for a response from the API) to the total page load and display time.

RESULTS AND DISCUSSION

For the practical application of the mathematical model developed on the basis of Amdahl's Law, an important step was the quantitative determination of the proportion of code to be optimised (P), as well as the acceleration coefficient of this proportion after the

implementation of the corresponding method (*N*). The reliability of the forecast of the overall acceleration of the system directly depends on the accuracy of these parameters. In the example with partial rendering, the value = 0.4 was obtained as the average value based on several measurements, where the full rendering process took about 400 ms out of a total time of 1,000 ms.

The acceleration factor was calculated as the ratio of the execution time of the corresponding stage before and after the implementation of the optimisation. For example, for partial rendering, which reduced the visual update time from 400 ms to approximately 133 ms, the acceleration factor was:

$$N_A = \frac{400}{133} \approx 3$$

Similarly, for caching API requests, where the server response time decreased from 300 ms to 150 ms, the acceleration factor was estimated as:

$$N_B = \frac{300}{150} = 2$$

Thus, the values of parameters *P* and *N* were obtained empirically, based on instrumental measurement and timing analysis. These values were used in further calculations to evaluate both the individual effect of each optimisation and their combined effect. Example 1: Partial rendering + API request caching. Partial rendering reduces the need for a full page refresh, optimising $P_A = 0.4$ (40% of rendering time). Caching API requests reduces the waiting time for data from the server, optimising $P_B = 0.3$ (30% of request time). Partial rendering helps caching (since fewer requests are sent to the server), so $R = -0.1$.

The values are substituted into the formula (7):

$$S_{combined} = \frac{1}{(1-0.4-0.3+(-0.1\times0.4\times0.3))+\frac{0.4}{3}+\frac{0.3}{2}}.$$

The calculations yielded the following:

$$S_{combined} \approx 1.87.$$

Thus, when each optimisation is applied separately, the expected acceleration would be approximately 1.67-1.75 times, but due to the positive effect of interaction (caching enhances rendering efficiency), the overall acceleration increases to 1.87 times. This example demonstrated that partial rendering combined with API request caching provides a greater total performance gain than each method separately (1.87 × vs. ~1.7 ×). This synergistic effect has also been confirmed in the works of other authors. In particular, V. Jain (2022) noted that the simultaneous use of lazy loading and JS code splitting technologies reduced web page loading time by up to 40% – significantly more than expected from each technique separately. Thus, the results were consistent with the current case: optimisations that complement each other (partial interface updates and data caching) can provide an enhanced cumulative effect. Moreover, a study by J. John (2024) showed that effective caching alone significantly speeds up the performance of a web system, reducing latency and increasing server throughput. This is consistent with choice of caching as one of the methods: in Example 1, caching API requests reduced the time to retrieve data by approximately half, which is consistent with the researcher's conclusions about the impact of cache on system latency. Thus, the literature data confirmed that combining rendering optimisation (on the client side) with request caching (on the server side) is an effective approach to speeding up content display.

Example 2: JavaScript code optimisation + CDN usage. Minification and merging of JavaScript files optimised $P_A = 0.25$. Using a CDN reduces the loading time of static resources, optimising $P_B = 0.35$. Since both optimisations reduce the overall load on the network, the effect is partially overlapping, so $R = 0.2$ (some files are already cached, so minification does not have the full effect).

Calculation:

$$S_{combined} = \frac{1}{(1-0,25-0,35+(0,2\times0,25\times0,35))+\frac{0,25}{2,5}+\frac{0,35}{3}}.$$

Result:

$$S_{combined} \approx 1,55.$$

In other words, in this example, it was found that minification and concatenation of JavaScript files together with the use of CDN provide a smaller total gain (≈ 1.55×) than would be expected from the independent effects of each method. The reason is the partial overlap of effects: both methods are aimed at accelerating the delivery of static resources and reducing network delays, so their effects are partially duplicated. A similar phenomenon – a drop in growth when optimisations overlap – has been noted by other researchers. For example, M.N.Y. Utomo *et al.* (2025) combined the use of a distributed CDN and caching (via Varnish) in an experiment to improve the performance of a web platform. The result was a significant improvement: throughput increased by 175%, average response time was reduced by more than half (54%), and network latency was reduced by 82%. These figures confirmed the effectiveness of using CDN and caching simultaneously. However, the authors also noted that both methods improve similar aspects of the system's performance (the speed of content delivery to the user). The current results clarified this point: when optimisations affect the same segment of the loading time (in this case, the time of file transfer and loading), the performance gain from combining them may be less than the total sum of the gains. Accordingly, when developing an optimisation strategy, it is worth considering possible areas of

overlap between methods to avoid a situation where the resources invested have a decreasing effect (the phenomenon of diminishing returns). Example 2 shows just such a case of partial duplication.

Example 3: Code splitting + server caching. Code splitting involves dividing the web application code into smaller modules that are loaded only when needed (for example, when a user accesses a specific section or function). Server caching reduces the time it takes to respond to requests, as repeat requests are processed using data already prepared in the cache on the server side. In the proposed scenario, it is assumed that: Code splitting covers 30% of the code (= 0.3) with an acceleration factor of = 3. Server caching affects 25% of operations (= 0.25) and provides a fourfold gain = 4). The interdependence coefficient between optimisations is = −0.05, which indicates mutual reinforcement: reducing the total code volume reduces the number of requests that require caching, and the presence of a cache, in turn, speeds up the loading of split modules. The total acceleration is calculated using the already known formula (7). The corresponding values are substituted as follows:

$$S_{combined} = \frac{1}{(1-0.3-0.25+(-0.05\times0.3\times0.25))+\frac{0.3}{3}+\frac{0.5}{4}}.$$

Calculations give approximately:

$$S_{combined} \approx 1.64.$$

Separate use of code splitting would speed up the system by approximately 1.25 times, and server-side caching alone by about 1.23 times. Thanks to the negative interdependence coefficient ($R < 0$), these two optimisation methods complement each other and provide a noticeably better cumulative result. In cases where $R > 0$ (for example, if parts of the functionality or data are duplicated, creating additional load), the overall acceleration could decrease due to mutual overlap of effects.

Thus, the combination of code splitting and server caching can yield a significant synergistic gain, but achieving optimal results depends on a thorough analysis of each method's impact on different parts of the code, as well as considering their degree of interdependence. The results obtained indicate that the overall gain from combining several optimisations depends not only on the proportion of code that can be accelerated but also on their degree of interdependence. A negative interdependence coefficient ($R < 0$) can ensure a better cumulative effect if optimisations complement each other and do not duplicate functionality. At the same time, a positive coefficient ($R > 0$) can reduce the final acceleration when optimisation methods partially overlap. Taking these conclusions into account, more effective optimisation strategies can be developed, aligning the nature of interaction between the methods used. This, in turn, allows for predicting potential performance gains and rationalising the costs of implementing web application performance improvement solutions.

Thus, the overall acceleration is ~ 1.64×, while the separate application of code splitting would accelerate rendering by ~ 1.25×, and server caching would reduce response generation time by ~ 1.23×. In other words, these methods complemented each other better than initially expected. A similar conclusion can be found in the work of J. van Riet & T.A. Ghaleb (2023), which described the sequential optimisation of a large web application by implementing 13 different improvements over a period of four months. As a result of this comprehensive approach, the first contentful paint metric improved by almost 98% (for the desktop version of the site), and other performance metrics also improved significantly. This case clearly demonstrated that combining several optimisation techniques (including those similar to those presented in this study: caching, resource loading optimisation, etc.) leads to a significant acceleration of the web application. In addition, the review by H.O. Ekpobimi *et al.* (2024) emphasised the need for an integrated approach: the authors noted that the simultaneous implementation of front-end optimisation strategies (such as code splitting, lazy loading, script minimisation) together with back-end improvements (different levels of data caching) allows for the greatest overall efficiency. The data obtained confirmed this thesis using the example of combining code splitting and server caching.

In the study by L. Kvurt & L. Tsyhylyk (2009), Amdahl's Law was considered a fundamental principle for evaluating the potential acceleration of systems through the optimisation of their components. Its application allows for theoretically determining the ultimate effectiveness of any optimisation, taking into account the limitations imposed by the unoptimised portion of the code. Importantly, to quantitatively predict such an effect, a model based on Amdahl's Law was applied, which considers the proportion of optimised code P and the acceleration N of each method, as well as their interdependence coefficient R. A similar modelling approach was proposed by C. Poolla & R. Saxena (2022), who extended the classical Amdahl's Law for the case of several simultaneous improvements and confirmed its validity with experimental data. The obtained results aligned with the conclusion that mathematical prediction of multifactor acceleration is possible and quite accurate. The distinction of the current study lies in the introduction of the R parameter to assess the nature of optimisation interaction. This allowed for explaining why in some cases (Examples 1 and 3) a greater than additive gain (synergy, $R < 0$) was observed, while in others (Example 2) – less (overlap effect, $R > 0$). This approach has not yet received widespread coverage in works, where most attention has been paid either to single optimisations or their empirical combination.

M. Hevery (2025) examined the application of Amdahl's Law in JavaScript code optimisation. The author

emphasised the importance of focusing on optimising those parts of the code that significantly affect the overall application performance, instead of expending resources on less significant fragments. The interaction between modern processors and virtual machines in the context of JavaScript execution was also discussed. Although the author provided valuable insights into JavaScript optimisation, the work did not propose specific mathematical models for evaluating the effectiveness of optimisations. The development of such models will allow for quantitatively assessing the impact of optimisations on web interface performance, which is critical in multi-project environments. Therefore, the contribution of this work lies in confirming the presence of both positive interaction and conflicts between different acceleration methods, as reported in previous studies, and in proposing a formalised method for evaluating this phenomenon. This has enabled developers to evaluate in advance the potential effect of combining several optimisations and to make informed decisions regarding web application performance improvement based on both theoretical calculations and best practices known from the literature.

CONCLUSIONS

Amdahl's classic law does not take into account the interaction between several optimisations, which makes it impossible to apply it directly to assess the cumulative acceleration during the simultaneous implementation of different methods. The proposed generalised model made it possible to assess the cumulative increase in performance, taking into account the interdependence between optimisations. The interdependence coefficient R, added to the model, made it possible to determine whether the optimisation methods interact synergistically and reinforce each other, or whether they overlap and reduce the overall acceleration. The model can be used for automated analysis of web application performance in the early stages of design, which contributes to a more rational allocation of resources and a well-founded choice of interface acceleration strategies.

The study made it possible to adapt Amdahl's law to take into account asynchronous requests, multi-level caching, and dynamic resource loading. This integration makes the performance assessment of web interfaces more accurate, as it covers specific aspects of modern technologies. The generalised model of combined optimisations considered several acceleration methods at once and made it possible to calculate their combined effect, taking into account possible interactions, which is reflected in the R coefficient. If the methods do not complement each other, the overall gain may be lower than the sum of the individual effects, while their synergy, on the contrary, can increase the final performance indicator.

Empirical examples have shown that combining caching with partial rendering can enhance the result by reducing the number of network calls and page load times. In this case, code splitting covers 30% of the code ($P_A = 0.3$) and provides acceleration with a coefficient of $N_A = 3$. Server caching affects 25% of operations ($P_B = 0.25$), resulting in a fourfold increase in performance ($N_B = 4$). The interdependence coefficient between these optimisations is $R = -0.05$. On the other hand, combining script minification with CDN may result in a more modest increase, as both methods partially overlap in their impact on network operations. The proposed formalised approach made it possible to predict such scenarios in advance and select the most appropriate optimisation strategies for each specific type of application (high-load service, content site, SPA, etc.).

The practical value of the results obtained lies in the ability to predict the effectiveness of web application optimisations before their implementation, as well as in a better understanding of the interaction of different approaches to acceleration. Web developers and DevOps engineers can make informed choices about the best strategies for improving the interface, avoiding unnecessary expenditure of resources on methods that do not provide significant improvement or conflict with each other. Further research should focus on verifying the models in real-world web system operating conditions, particularly with large amounts of data and involving large categories of users. This will provide additional evidence of the correctness and universality of the developed approach, as well as contribute to its further development and improvement.

ACKNOWLEDGEMENTS

None.

FUNDING

None.

CONFLICT OF INTEREST

None.

REFERENCES

[1] Abounacer, R., Afdel, K., & Bouaouda, A. (2023). Resource utilization and cost implications of container live migration in clouds: An approach performed on Amazon Web Services (AWS). *Research Square*. doi: 10.21203/rs.3.rs-3286731/v1.

[2] Ahmed, F., Erman, J., Ge, Z., Liu, A. X., Wang, J., & Yan, H. (2017). Detecting and localizing end-to-end performance degradation for cellular data services based on TCP loss ratio and round trip time. *IEEE/ACM Transactions on Networking*, 25(6), 3709-3722. doi: 10.1109/TNET.2017.2761758.

[3] Araújo, G.R., Gomes, R., Ferrão, P., & Gomes, M.G. (2024). Optimizing building retrofit through data analytics: A study of multi-objective optimization and surrogate models derived from energy performance certificates. *Energy and Built Environment*, 5(6), 889-899. doi: 10.1016/j.enbenv.2023.07.002.

[4] Dyvak, M., & Kindzerskyi, O. (2024). Investigation of the efficiency of parallel computational scheme for identification of interval discrete models based on swarm intelligence. *Herald of Khmelnytskyi National University. Technical Sciences*, 331(1), 29-37. doi: 10.31891/2307-5732-2024-331-3.
[5] Ekpobimi, H.O., Kandekere, R.C., & Fasanmade, A.A. (2024). Conceptual framework for enhancing front-end web performance: Strategies and best practices. *Global Journal of Advanced Research and Reviews*, 2(1), 99-107. doi: 10.58175/gjarr.2024.2.1.0032.
[6] Felani, R., Al-Azam, M.N., Adi, D.P., & Widodo, A. (2020). Optimizing virtual resources management using Docker on cloud applications. *Indonesian Journal of Computing and Cybernetics Systems*, 14(3), article number 319. doi: 10.22146/ijccs.57565.
[7] GeeksForGeeks. (2023). *Computer organization. Amdahl's law and its proof*. Retrieved from https://www.geeksforgeeks.org/computer-organization-amdahls-law-and-its-proof.
[8] Hevery, M. (2025). *Amdahl's law. Bare metal JavaScript: The JavaScript virtual machine*. Retrieved from https://frontendmasters.com/courses/javascript-cpu-vm/amdahl-s-law.
[9] Jain, V. (2022). Optimizing web performance with lazy loading and code splitting. *International Journal of Core Engineering & Management*, 7(3), 193-199. doi: 10.5281/zenodo.14956631.
[10] John, J. (2024). *Optimizing application performance: A study on the impact of caching strategies on latency reduction*. Retrieved from https://www.researchgate.net/publication/385916660_OPTIMIZING_APPLICATION_PERFORMANCE_A_STUDY_ON_THE_IMPACT_OF_CACHING_STRATEGIES_ON_LATENCY_REDUCTION.
[11] Junghans, C., Agarwal, A., & Delle Site, L. (2017). Computational efficiency and Amdahl's law for the adaptive resolution simulation technique. *Computer Physics Communications*, 215, 20-25. doi: 10.1016/j.cpc.2017.01.030.
[12] Kothapalli, M. (2022). Performance analysis of single page applications. *International Journal of Science and Research (IJSR)*, 11(1), 1631-1635. doi: 10.21275/SR24529184457.
[13] Kundos, M.H., Solovey, L.Ya., Hrysyuk, A.V., & Bahniuk, O.M. (2024). Efficiency and multi-threading of parallel calculations in systems programming. *Taurida Scientific Herald. Series: Technical Sciences*, 5, 60-64. doi: 10.32782/tnv-tech.2024.5.6.
[14] Kvurt, L., & Tsyhylyk, L. (2009). The use of Amdahl's and Gustafson's laws in evaluating the acceleration factor in multiprocessor systems. *Measurement Techniques and Metrology*, 70, 55-56.
[15] Ma, J. (2024). A high performance computing web search engine based on big data and parallel distributed models. *Informatica*, 48(20), 27-38. doi: 10.31449/inf.v48i20.6776.
[16] Majchrzak, T.A., Biørn-Hansen, A., & Grønli, T.-M. (2018). Progressive web apps: The definite approach to cross-platform development? In *51st Hawaii international conference on system sciences (HICSS 2018)* (pp. 1-10). Hawaii: Hilton Waikoloa Village. doi: 10.24251/HICSS.2018.718.
[17] Malavolta, I., Chinnappan, K., Jasmontas, L., Gupta, S., & Soltany, K.A.K. (2020). Evaluating the impact of caching on the energy consumption and performance of progressive web apps. In *Proceedings of the IEEE/ACM 7th international conference on mobile software engineering and systems* (pp. 109-119). New York: Association for Computing Machinery. doi: 10.1145/3387905.3388593.
[18] Mitton, L. (2023). *Amdahl's law: Understanding the basics*. Retrieved from https://www.splunk.com/en_us/blog/learn/amdahls-law.html.
[19] Nair, A.M., Sivaiswarya, C.K., Sidharth, S., Visakh, K.K., & Joy, J. (2024). Dockerized application with web interface. *International Journal of Scientific Research in Computer Science Engineering and Information Technology*, 10(2), 412-419. doi: 10.32628/CSEIT243646.
[20] Oh, S., Kwon, Y., & Lee, J. (2025). Optimizing real-time object detection in a multi-neural processing unit system. *Sensors*, 25(5), article number 1376. doi: 10.3390/s25051376.
[21] Poolla, C., & Saxena, R. (2022). On extending Amdahl's law to learn computer performance. *ArXiv*, 2110, article number 07822. doi: 10.48550/arXiv.2110.07822.
[22] Potdar, A.M., Narayan, D.G., Kengond, S., & Mulla, M.M. (2020). Performance evaluation of Docker container and virtual machine. *Procedia Computer Science*, 171, 1419-1428. doi: 10.1016/j.procs.2020.04.152.
[23] Prus, O.V., Maidaniuk, V.P., & Arseniuk, I.R. (2024) March analysis of tools for multiproject environments management: Optimization of the software development. *Scientific Works of VNTU*, 1, 29-36. doi: 10.31649/2307-5376-2024-1-29-36.
[24] Ren, J., Gao, L., & Wang, Z. (2024). JavaScript performance tuning as a crowdsourced service. *IEEE Transactions on Mobile Computing*, 23(5), 6116-6132. doi: 10.1109/TMC.2023.3316167.
[25] Utomo, M.N.Y., Tungadi, E., & Khartika, W. (2025). Enhancing web performance for e-learning platform using content delivery network (CDN) and varnish cache. *Journal of Information Systems and Informatics*, 7(1), 831-847. doi: 10.51519/journalisi.v7i1.993.
[26] van Riet, J., Ghaleb, T.A. (2023). Optimise along the way: An industrial case study on web performance. *Journal of Systems and Software*, 198, article number 111593. doi: 10.1016/j.jss.2022.111593.
[27] Vepsäläinen, J., Hellas, A., & Vuorimaa, P. (2024). Overview of web application performance optimization techniques. *ArXiv*, 2412, article number 07892. doi: 10.48550/arXiv.2412.07892.

Оптимізація продуктивності веб-інтерфейсів із використанням закону Амдала

Олег Прус
Аспірант
Вінницький національний технічний університет
21021, вул. Хмельницьке шосе, 95, м. Вінниця, Україна
https://orcid.org/0009-0007-2514-9871
Володимир Майданюк
Кандидат технічних наук, доцент
Вінницький національний технічний університет
21021, вул. Хмельницьке шосе, 95, м. Вінниця, Україна
https://orcid.org/0000-0002-3345-2578

Анотація. У статті розглянуто побудову та застосування узагальнених математичних моделей на основі закону Амдала для визначення максимально можливого прискорення роботи веб-інтерфейсів з урахуванням їхніх ключових особливостей. Запропоновано розширення класичного підходу шляхом включення до моделі асинхронних процесів, механізмів різнорівневого кешування та методів динамічного завантаження ресурсів, що дає змогу точніше оцінювати сумарний вплив різних оптимізацій на швидкодію. Зокрема, обґрунтовано доцільність урахування асинхронного обміну даними, який дає змогу обробляти запити паралельно й уникати блокувань у процесі оновлення контенту. Розроблено формулу, що враховує ефективність клієнтського і серверного кешу та надає кількісну оцінку скорочення часу відгуку при повторному використанні вже завантажених даних. Особливу увагу зосереджено на методиках покрокового отримання вмісту, коли початкове завантаження сторінки мінімізується за рахунок відкладеного додавання окремих скриптів, зображень або стилів, що дає змогу пришвидшити початкове відображення важливого контенту і зробити інтерфейс більш чутливим до дій користувача. Окрім того, розглянуто вплив комплексного поєднання оптимізаційних стратегій на продуктивність веб-інтерфейсу та запропоновано відповідну узагальнену модель, яка за допомогою коефіцієнта взаємозалежності дозволяє визначити, наскільки одна оптимізація підсилює або, навпаки, нівелює дію іншої. Це надає можливість прогнозувати сумарний приріст швидкодії та зіставляти витрати на впровадження кількох рішень із потенційною економією часу. Запропонований формалізований підхід може стати основою для створення автоматизованих засобів оцінки продуктивності веб-інтерфейсів, інтегрованих у процес розробки. Перевірка моделі в трьох практичних сценаріях – частковий рендеринг з кешуванням API, мініфікація JavaScript з content delivery network (CDN) та розділення коду з кешуванням на стороні сервера – дала приріст продуктивності відповідно в 1,87 ×, 1,55 × та 1,64 ×, що повністю відповідає теоретичним прогнозам. Отримані дані підтверджують здатність коефіцієнта взаємозалежності R точно відображати синергію або накладання ефектів оптимізацій і роблять модель придатною для попереднього вибору найефективніших стратегій прискорення на етапі CI/CD-аудиту

Ключові слова: математичні моделі; асинхронне виконання; кешування даних; динамічне завантаження; комбіновані стратегії; прогнозування швидкодії