Synchronous Vs Asynchronous Content Loading On The Internet

Miloš Mladenović
Academy of Professional Studies Sumadija
Department in Trstenik,
Trstenik, Serbia
milosmladenovic037@gmail.com

Marija Mojsilović
Academy of Professional Studies Sumadija
Department in Trstenik,
Trstenik, Serbia
mmojsilovic@asss.edu.rs

Abstract—**Fast and fluent loading of content on the Internet today is becoming an imperative of modern web programming, which makes IT today, faster than ever, developing and expanding its domain of activity, in order to provide users with the most comfortable and pleasant use of Internet content. Traditional approaches in programming provide the main support for the development of new technologies and approaches in solving the more complex problems that engineers face today. The ever-increasing amount of data, which must be processed, converted and made available to the user in a form that represents easy data manipulation in the shortest possible time, requires that the asynchronous model of loading content on the Internet replaces the previous sequential-synchronous model and that it be improved implements in the daily operation of applications, programs, websites and the like. Accordingly, the goal of this work is to explain how the asynchronous, or synchronous, content loading model works, as well as to prove that the asynchronous content loading model on the Internet is superior to the synchronous model. Therefore, the subject of this paper is synchronous vs. asynchronous loading of content on the Internet**

Keywords- synchronous content loading; asynchronous content loading; web loading; web page rendering; rendering; on-screen loading

I. INTRODUCTION

The modern society of the 21st century implies an accelerated pace of life and the rapid progress of technology. Therefore, the progressive development of hardware and software does not allow the user, who is used to getting everything "*now and immediately*", to wait in order to access the content that interests him. And so in the world of Information Technologies, and especially in the areas of web programming, the aspiration has been redirected to finding a more efficient solution for fast loading of web pages, that content which represents the goal of interest and demand of the user himself. One of the important conditions that should not be neglected is the efficiency of the programming itself, the way of manipulating the data that is loaded. Also, a significantly greater impact, with more complex web applications, today when time is money, when the user does not want to have the feeling that he has to wait for the content of the web page to load; it is essential to think about the visual-aesthetic side of content loading. Using asynchronous access to the content of web applications in 2022 is considered the best practice so far. In the past, the traditional, sequential approach worked by stopping the application, and the user's screen "***freezes***" until the content is not available for manipulation by the user and the application starts working again. Today, such an approach to loading web pages is considered a not so good solution, although we can still encounter such a way of working applications and accessing a certain site and web page. Therefore, asynchronous access to the content of the web application not only enables the timely appearance of the searched content on the screen, whether it is browsing or accessing the site and the application via mobile phones, tablet devices or computers, which the asynchronous method of programming offers as a solution, because a large amount of data from the database needs to access external application programming interfaces in the shortest possible time, which on the other hand allows the server in the background of the application not to depend on the results of some slow operation, until that same operation is performed. It strives for functionality in providing services to other programs or devices called clients, all for the purpose of satisfying the user himself in interactive communication with certain web content. The aim of this work is to prove that the asynchronous model of content loading on the Internet is superior to the synchronous model.

II. PREVIOUS RELATED RESEARCH

The traditional, existing synchronous programming model relies on operations that are performed one after the other, that is, sequentially, and the program moves to the next step only after the previous step has been fully completed. Such a linear approach to the execution of tasks encounters great difficulties and problems. Let's take into account that in modern client applications, the need for constant interaction with available searchable content, which is located outside the working memory, is becoming more and more frequent, which the synchronous programming model, and then the result of loading content from the Internet, cannot satisfy, because a longer period of time is required to perform all the operations one after the other. An example of this way of executing

instructions with synchronous system calls blocks the thread of execution of loading content until, for example, reading from disk is complete. This type of programming tends to be solved by the principle of "**Multithreading**", where it is possible to perform several parallel processes at the same time, and whose program work relies on an asynchronous programming model that allows several operations to be performed at the same time. Which provides greater comfort to the user when loading content from the Internet [1].

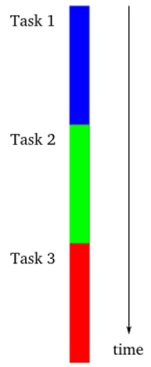

Figure 1. The single-threaded synchronous model [2]

Figure 2. The asynchronous model [2]

III. Asynchronous Rendering of Website

Asynchrony in programming implies the processing of events that can happen outside the main flow of execution, i.e. such a processing mechanism that does not depend on the management of the main flow "***Control flow***", usually such events are interprocess signals or instigating actions of the program that will execute during the execution of the main program without any blocking of the Internet page while waiting for the results of the search itself [3]. More precisely, during asynchronous programming, the program is free to issue commands addressed to the database or network devices in parallel while the processor continues to work and execute the given commands. This fluent approach in programming is especially useful in the case of solving problems related to input and output "***I/O bound***", where the execution time is actually the waiting time for the results of the requested external application programming interface, when we talk about web programming the waiting time is in matters query time over the database. Practically, if we look at the server side, this kind of asynchronous model offers us the possibility that while we wait for the data of one client, we do not ignore the requests of another client, but at that moment the capacities of the processor will be used to the maximum, so that the waiting time for loading the website and content is minimal . For the user himself, this means that he is enabled to display and respond to the desired interaction with the content, while the application is constantly running in the background. However, although it seems that this approach is far better and more efficient than the traditional synchronous model, it is also not so efficient when it comes to programs that are tied to the processor "***CPU bound***", because then the waiting time, i.e. execution of a command determined by the time it takes the processor to process a large amount of data. Therefore, the asynchronous approach requires additional effort in order to match the client interface with the data, so it is largely necessary to block or match entire parts of the interface from the server to the final destination of displaying the content, i.e. displaying the processed data [4].

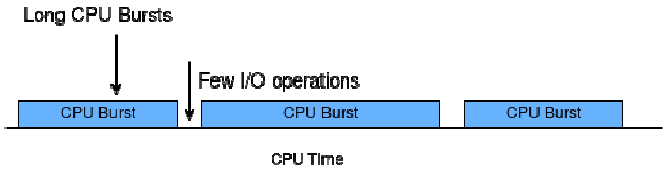

Figure 3. CPU – IO Bound (source: https://www.baeldung.com/cs/cpu-io-bound

A. *How to speed upweb page loading?*

Since we are already talking about loading content from the Internet, the logical question is how to speed up the loading of web pages. Therefore, sites that strive for high user performance require the programmer to devise the sequence in the code in advance and accurately predict the places where styles and scripts will be called, so calling external **CSS** at the bottom of the page generally slows down its loading, and its movement in The *HEAD* part would permanently solve such a problem. However, when we talk about scripts, external **JavaScript** files cause a similar problem, but the solution to such a problem would be completely opposite to the previous one, therefore, it is better to move the scripts from the top to the bottom of the page and as low as possible, and the reason for this movement is progressive loading i.e. page rendering, and therefore greater parallelism when downloading the necessary script files.

When dealing with external CSS, progressive loading is blocked until all styles are loaded. That's why it's best to move the style call to the *HEAD* part of the page, so that they are downloaded first. With scripts, progressive loading is blocked for all content below the script. Moving the script as low as possible means that the content above the script will be loaded first.

Another problem caused by scripts is blocking parallel downloads. The HTTP/1.1 specification suggests that browsers download no more than two components in parallel per host. If for example images are uploaded from multiple hosts, it is possible to get multiple downloads in parallel. However, while the scripts are being downloaded, the browser will not start any other downloads, even on different hosts.

In some situations it is not easily feasible to move the images to the bottom. If, for example, a script uses *document.write* to load part of the page's content, then the script cannot be moved down. There can also be range issues. In many cases there are ways to resolve these situations.

An alternative suggestion that comes into play is the use of "***deferred scripts***". The DEFER attribute indicates that the scripts do not contain *document.write*, which signals the browser to continue loading the page. Unfortunately, Firefox does not support the DEFER attribute. In Internet Explorer, scripts can be delayed, but not completely as you would like.

If the script can be delayed, that means it can also be moved to the bottom of the page. These are some of the tips on how to speed up the loading of web pages [5]. Optimize images and video content, because over 40% of web page content consists of images and photos. It is considered that the optimal size of photos is up to 150KB and that there is no need to display images of huge resolutions, because just waiting for them to be downloaded by users, if they have a bad internet connection, can only spoil the user experience of using your site.

It is also recommended to test the speed and performance of the site itself after creating the website or during development. If you follow the previously defined guidelines, you will surely improve the loading speed of your site.

IV. Server-side Rendering

A term often used when loading content is "**server-side rendering**", which is actually the process of asynchronously rendering a web page into plain HTML before it is sent to the client. This process used to be more prevalent than today, but it still exists today, especially with programming languages such as **PHP**, where data is fetched from the database, compiled into an HTML document and thus sent to the user [6].

In this way, predefined content is displayed to the user most often in the form of dynamic web pages. After a certain time interval, the content that will be displayed to the user may be different compared to the previous visit. In contrast to dynamic web pages, there are static ones where the name itself indicates that the content of that page is static and does not change.

TABLE I. Server Vs. Client-side page rendering based on freeCodeComp.org [7]

	Server-side	Client-side
Cons (+)	• Search engines can crawl the site for better SEO. • The initial page load is faster. • Great for static sites.	• Low SEO if not implemented correctly. • Initial load might require more time. • In most cases, requires an external library.
Pros (-)	• Frequent server requests. • An overall slow page rendering. • Full page reloads. • Non-rich site interactions.	• Rich site interactions • Fast website rendering after the initial load. • Great for web applications. • Robust selection of JavaScript libraries.

Acknowledgment (Heading 5)

In order to prevent the creation of a hell of callback functions, there was a need to introduce more advanced elements of the JavaScript language. The **ES6 standard** introduced the concept of **promises**, which solves this problem.

A promise can be in one of the following states [8]:

- *Pending* — The promise is in this state during initialization.
- *Fulfilled* — The promise moves to this state if the asynchronous operation has completed successfully.
- *Rejected* — The promise goes into this state if the asynchronous operation ended unsuccessfully.

Now that we're familiar with a popular way to manage asynchronous code using promises, we can show a sample code that uses the JavaScript language. The ECMAScript2017 standard provides a way to write asynchronous functions, where they look like synchronous functions, but of course, behave asynchronously, using the async and await keywords [8, 9].

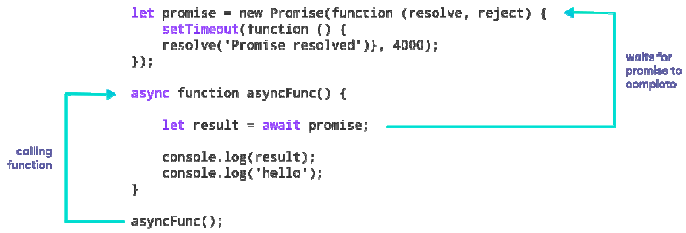

Figure 4. An example of asynchronous JavaScript (source: https://www.programiz.com/javascript/async-await

This way of calling functions can be used as one of the main ways of presenting content to users and is the very foundation and basis that can be expanded for better adaptation for the specific application in which it will be called. The following is a code example that shows how content can be displayed with asynchronous loading.

HTML document with asynchronous content loading

```
<!DOCTYPE html>
<html>
<body>
  <h2>JavaScript async / await example</h2>
  <p id="page-content"></p>
  <script>
    async function renderPage() {
      let myPromise = new Promise(function (resolve) {
        let req = new XMLHttpRequest();
        req.open('GET', "page-content.html");
        req.onload = function () {
          if (req.status == 200) {
            resolve(req.response);
          } else {
            resolve("File not Found");
          }
        };
        req.send();
      });
        document.getElementById("page-content").innerHTML =
await myPromise;
    }
    renderPage();
  </script>
</body>
</html>
```

V. Experimental Results

In 2019, the research "*Does asynchronous model really give benefits in throughput against properly configured synchronous?*" was conducted by Russian software engineer Eugene D. Gubenkov. In that research, Eugene created two ASP.NET Web API methods that he tested using Jmeter, a

tool for testing, analyzing and measuring the performance of various services, with a focus on web applications.

You will notice how the synchronous model becomes faster when the thread pool injects enough threads. If the Thread Pool is set correctly, this way would be better than asynchronizing from the beginning the beginning.

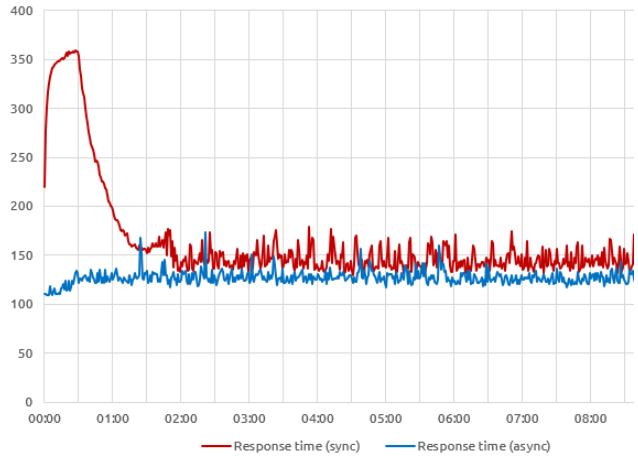

Figure 5. Response time with thread count [10]

If we wonder what happens with resource consumption, under the claim "***Thread has big cost in terms of CPU time scheduling, context switching and RAM footprint***". And not everything is like that. Thread scheduling and context switching is efficient. In terms of stack usage, a thread does not immediately consume RAM, it just reserves the virtual address space and commits only the small part that is actually needed.

Let's see what the data says. Even with more threads the sync version has a smaller memory footprint (the working set mapped to physical memory).

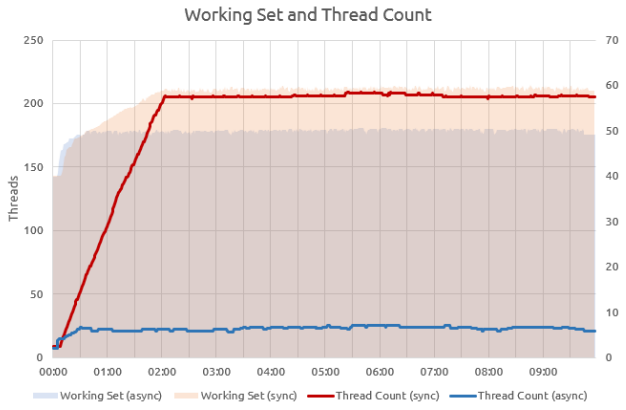

Figure 6. Working Set and Thread Count [10]

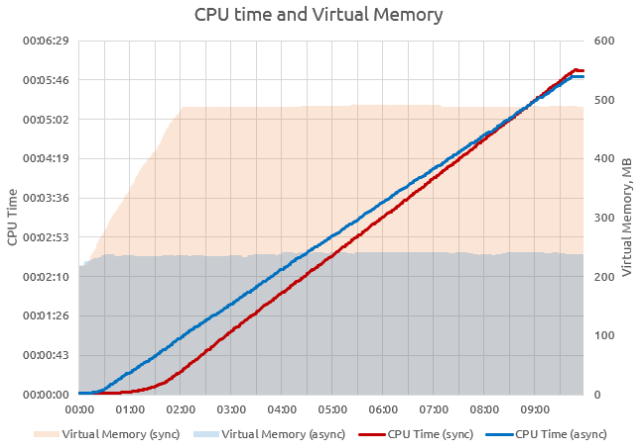

Figure 7. CPU time and Virtual Memory [10]

So we conclude that under some laboratory or specific circumstances it is possible to obtain comparable results for synchronization versus asynchrony, but in real cases where the workload cannot be 100% predictable and where the workload is uneven, it is inevitable that we reach some kind of limitation threads, or server-side limits, or thread pool growth limits. Additionally, the sync version has a larger memory footprint (both working set, and way bigger virtual memory size). In terms of CPU consumption, the asynchronous model also wins [10].

CONCLUSION

When we talk about technologies of this kind, especially when there is an increasingly present emphasis on web programming, we should emphasize the importance of the very availability and modernity of today, which requires the implementation of modern knowledge of Information Technology in what it creates, so that the users of a newly built application or site, web page provided everything that the demanding user demands and expects from it today, while at the same time fulfilling and relying on scientific and technological requirements at all times and being methodologically set, and at the same time commercial and visible to users, i.e. adapted for wider use and the possibility of further development and improvement, which reflects the importance of web programming. The path of transition that modern society is going through and the increasing digitization and automation, as well as data manipulation, have led to the fact that web technologies and the programming languages and design that would support them have become new frameworks in the marketing of web content on the information technology market. The content should be appropriate, direct, quickly and efficiently placed, easily accessible and visible, which speaks in favor of the fact that the asynchronous method of loading content from the Internet is in a leading position in the selection of models and methods of programming, which at the same time provides a lot of space for further upgrading and improving all necessary performance. The 21st century does not suffer from a delay in loading content, but requires fluency, availability and

applicability at a high level, and at the same time a satisfied user.

Future Scope

How can we predict the internet future, when it changes from second to second, what is current now and seems to be the most acceptable and most rational, most innovative solution in programming and the very selection of methods and techniques, languages and programs, already at this moment, while you are reading these sentences, it becomes an outdated way of thinking and some new solutions have already been created. It is precisely in this dynamic and the power of transformation that Information Technologies are reflected, which are always a step or three ahead of what is currently marketed, just like the asynchronous model of uploading content to the Internet. Although we think that the entire process of reading the content is finished, actually in the background the server is still working and creating the new content that is shown to us. The user follows the trends and claims that his use of the Internet is futuristic, but he does not really understand how information technology, artificial intelligence and the way of creating the future are actually ahead of the current future. Therefore, it is not possible to speak with precision about the future trends that will be imperative in solving the problem, but perhaps it is best to speak about the proposals of future achievements, and accordingly, as we conclude, progressive web applications with an asynchronous loading model are leading the way and will definitely mark the future programming.

References

[1] Andić A., (2021), Upravljanje stanjima unutar React biblioteke – Master's thesis, Pula, University of Pula

[2] cs.brown.edu, *Introduction to Asynchronous Programming*, https://cs.brown.edu/courses/cs168/f12/handouts/async.pdf, [Viewed 25 August 2022]

[3] Davies A., (2012), Async in C# 5.0: Unleash the Power of Async, Sebastol

[4] Marić S., (2020), Razvoj asinhronih veb aplikacija korišćenjem radnog okvira Tornado – Master's thesis, Belgrade, University of Belgrade, Faculty of Mathematics

[5] Ivezić J., *Kako Ubrzati Učitavanje Web Stranica*, https://www.popwebdesign.net/popart_blog/2013/05/kako-ubrzati-ucitavanje-web-stranica/, [Viewed 26 August 2022]

[6] Mihajlović N., *Renderovanje na strani servera – da li je i dalje potrebno?*, https://www.mcloud.rs/blog/renderovanje-na-strani-servera-da-li-je-i-dalje-potrebno/, [Viewed 27 August 2022]

[7] Vega C., Client-side vs. server-side rendering: why it's not all black and white, https://www.freecodecamp.org/news/what-exactly-is-client-side-rendering-and-hows-it-different-from-server-side-rendering-bd5c786b340d/, [Viewed 27 August 2022]

[8] Ajzenhamer N., Zečević A., (2021), *Programiranje za veb*, Belgrade, University of Belgrade, Faculty of Mathematicsž

[9] programiz.com, *Javscript async/await*, https://www.programiz.com/javascript/async-await, [Viewed 28 August 2022]

[10] Gubenkov D. Eugene, *Does asynchronous model really give benefits in throughput against properly configured synchronous?*, https://stackoverflow.com/questions/55823184/does-asynchronous-model-really-give-benefits-in-throughput-against-properly-conf , [Viewed 28 August 2022]