Analyzing the role of CDNs in web traffic

Hannes Koksch

Summer term 2024

This scientific paper was written as part of the lecture *Internet Traffic, Performance and Content Distribution* at *Hochschule der Medien Stuttgart*.
All assets belonging to this work can be found at:
https://github.com/hanneskoksch/Analysis-of-CDNs

1 Introduction and goal

The digital landscape has evolved to the point where speed, reliability, and scalability of web content delivery are crucial for the user experience. Content Delivery Networks (CDNs) play an important role in ensuring that web content reaches users efficiently regardless of their geographical location. CDNs achieve this by distributing content across a network of servers located around the world to minimize latency and load times. According to the 2022 Web Almanac report, 29% of sites use a CDN to serve their HTML and 47% of sites serve resources from subdomains using a CDN [2]. This widespread adoption demonstrates their importance, but it also raises interesting questions about their current market share and impact on overall web traffic.

The purpose of this paper is to analyze the use of CDNs by the top 1000 websites, focusing on the distribution of requests handled by different CDNs, with the goal of gaining an understanding of what the market for web content delivery using CDNs looks like, how CDNs can be identified, and also to get a vague forecast of the next Web Almanac report.

Therefore, the following research questions were pursued:

1. Which CDNs are used by the top 1000 websites?
2. How often are they used?
3. What percentage of the total delivered website assets are served by CDNs?

In section 2, a discussion is held about choosing the dataset to analyze the top 1000 websites and the CrUX dataset is selected. Section 3 outlines the methodology used to capture and analyze CDN requests, detailing the tools and techniques. In section 4, the paper presents the findings on the distribution and impact of CDNs on web traffic and answers the research questions. Finally, section 5 provides a critical review of the results and discusses the implications of the findings.

2 Choosing the dataset

For the purpose of analyzing the "top 1000" websites, a decision has to be made for a dataset. There is no one correct list, but a variety of lists offered by SEO companies, web tracking organizations, browser vendors, search engines, or DNS server providers. Due to the distributed nature of the Internet's architecture, none of these can technically provide a complete picture, and all lists have their biases. Also, interpretations of the "top 1000" sites may differ from popularity as measured, for example, by server requests to the site ranking of a well-established search engine.

In the past, many Internet measurement papers used the popular *Alexa* dataset, which was shut down by Amazon in 2021 [11]. The measurements in this paper therefore rely on the Chrome User Experience Report (CrUX) dataset. This contains data collected by the widely used Chrome web browser. According to Ruth et al. 2022 [8], the CrUX is the most accurate toplist compared to Cloudflare's own metrics and is therefore the generally recommended dataset for measuring websites on the Internet. The CrUX dataset is also used by the Web Almanac [3].

The CrUX dataset does not come with an exact ranking, but with a classification into buckets (top 1000, top 5000, top 10000, ...) in which the data is randomly distributed. The dataset can be accessed via a public Google BigQuery database [7]. For the measurements of this paper, however, the already cached and downloadable BigQuery query on GitHub is used. [12]

The data provided in this dataset, and therefore the data used in this paper, may contain adult websites, gambling related content, or other material that may appear untrustworthy. Readers are advised to exercise caution and discretion when accessing or interpreting such content. The author is not responsible for the content or reliability of external websites, and the inclusion of such links does not imply endorsement or verification of the information. Please keep this in mind when reviewing any of the data, lists, or repositories mentioned in this work.

3 Methodology

To get from a dataset of websites to data that can be analyzed regarding their CDN usage, a few steps are necessary. These steps are *capturing the requests*, *identifying the CDNs*, and a subsequent *data analysis*. A set of Python tools was used for this purpose to keep everything in the Python environment. The following sections describe these three steps and their considerations in detail.

3.1 Mesurement metadata

For the measurement, the CrUX dataset from may 2024 was used and the data was last recorded on 10.08.2024.

3.2 Capturing the requests

To get more information about how a website's content is actually delivered, it is necessary to make a request to that website and capture its responses. This automated iteration through the CrUX dataset was done via a Python script using several additional tools, such as the Selenium WebDriver and Headless Chrome.

The Selenium WebDriver is a tool used to automate web browser interactions, allowing the script to visit websites and capture network traffic. It is typically used for testing and automating tasks and makes the W3C recommended web driver API accessible across multiple browsers [9].

By running Selenium with Headless Chrome, the browser interactions were performed in the background, making the process more efficient for large-scale data collection. After a loading time of 10 seconds for each website, Selenium captured all HTTP responses, including those for images, scripts, and other resources. This comprehensive capture is important for accurately identifying CDNs, as many resources are dynamically loaded after the initial page load. The 10 seconds was the reasonable compromise between loading time and data collection yield.

This simulated the browser behavior of a real user visiting the pages, and provided a significant advantage over simpler HTTP libraries that only retrieve the raw HTML of a page. The much simpler approach of just examining the HTML would not have detected any dynamic content, but only the HTML itself and the resources embedded directly in the file - so no dynamic API requests and fetch calls coming, for example, from JavaScript code or a whole framework that needs to be loaded and executed first. But it is still an approach that tools like the consulting site CDN Planet's CDN Finder[1] use to get a quicker and easier answer to which CDN a site is generally using that is lightweight enough to even run on the edge [6].

The captured data was written to a `.csv` file to make it accessible for the identification process and analysis. It contains for each request the originally requested page of the CrUX dataset, the full URL of the requested resource and snippets of it in a separate column, the IP address, the resolved Canonical Name (CNAME) record of the DNS, the Autonomous System Number (ASN) and its description, and the response headers. The CNAME was resolved using the Dnspython package with an equivalent of the `nslookup` command in the terminal to get the last entry of the CNAME chain. The ASN was resolved by mapping the IP address to an autonomous system using the *IP address to ASN database*[2] in its self-hosted version[3]. The resolving was done at the time of data collection, its meaning and use is described in the following section.

[1] `https://www.cdnplanet.com/tools/cdnfinder`
[2] `https://iptoasn.com/`
[3] `https://github.com/jedisct1/iptoasn-webservice`

3.3 Identifying the CDNs

Several measures were taken to identify the CDN usage of each request. Step by step, the response header, the URL, the CNAME, and finally, if those failed, the ASN was used as the last step to match as many requests as possible. This approach is also used by CDN Planet's CDN Finder [6].
The four steps are described in more detail in the list below:

1. **Response header:** Some CDNs send their own custom response header along with the requested resource. This makes it very easy to identify a CDN. A list of these headers has been manually collected and maps the headers to the actual names of the CDNs, with entries looking like this, for example `["x-amz-cf-id", "Amazon Cloudfront"]`. For this step, only the response headers had to be searched.

2. **URL:** CDNs have their own URLs that they use to deliver their contents. In cases where these URLs are not obscured, they can be used to identify the CDN behind the requested resource. The list that maps these URLs to the names of the CDNs was obtained from the GitHub repository turbobytes/cdnfinder[4], a deprecated open source version of the now closed source CDN Finder tool by CDN Planet already mentioned above. For this step, only the URL had to be searched.

3. **CNAME:** In case the URL is obscured using a CNAME record, checking the resolved hostname can reveal the actual CDN provider. To map the URLs to the names of the CDNs, the same list as with just the direct URL was used. For this step, the CNAME record had to be resolved beforehand and then also only the new URL had to be searched.

4. **ASN:** CDNs can be assigned to an ASN and its description based on their IP address. A list of the descriptions of the autonomous systems that maps these to the actual names of the CDNs has been manually collected and takes into account the top CDNs in the 2022 Web Almanac that were also appearing the most in the collected data. For this step, the autonomous system had to be resolved beforehand and then the descriptions had to be searched.

[4] `https://github.com/turbobytes/cdnfinder`

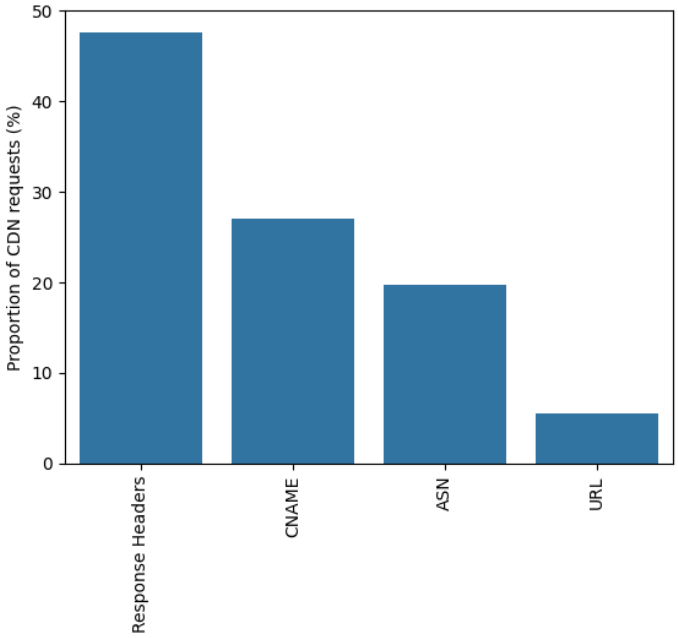

Figure 1: Ways of identification of CDNs

In figure 1, an advance look at the analysis findings shows that the response headers proofed to be the most helpful factor in identifying the CDNs with a proportion of just under 50%. Whereas checking only the URL, which was the second step, resulted in less than 10% hits. For this statement, only requests identified as being served by a CDN were considered.

3.4 Data analysis

In this last step, the collected and categorized data was aggregated and used to create charts that answer the research questions and also highlight some other interesting aspects. A Jupyter Notebook was created in which the analysis was done relying heavily on Pandas, a powerful library to handle and manage data, and Seaborn, a data visualization library used for creating charts. The findings of this analysis are described in the next section.

4 Findings

In this section the findings of this work are presented as a result of the analysis. With that the initial research questions are answered but also other interesting aspects are highlighted.

4.1 Share of CDNs

Dividing the data into the categories "Not using a CDN / not identified" and "Using a CDN" by pages shows, that just under 35% of pages use a CDN at all, as can be seen in figure 2. The disclaimer "not identified" was added because not every CDN might be detected though, as stated in more detail in section 5.

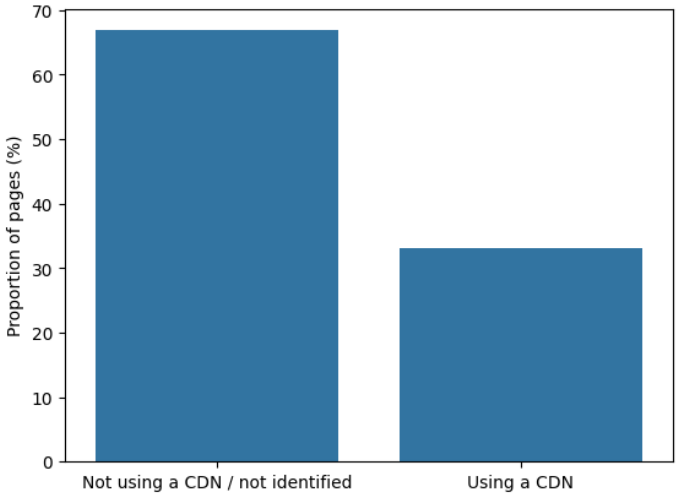

Figure 2: Share of CDNs by page

Taking a closer look at just the requests, however, figure 3 shows that about 77% of the requests were served by a CDN. This answers research question 3. At the same time, it can be concluded that although there are fewer sites that use a CDN, they require more website assets and therefore more requests overall.

From now on looking only at the CDN data, a comparison of the different CDN providers highlights six big providers with over 5% share each and with Cloudflare being by far the largest with more then 25%, as figure 4 shows. This answers research question 1 and 2.

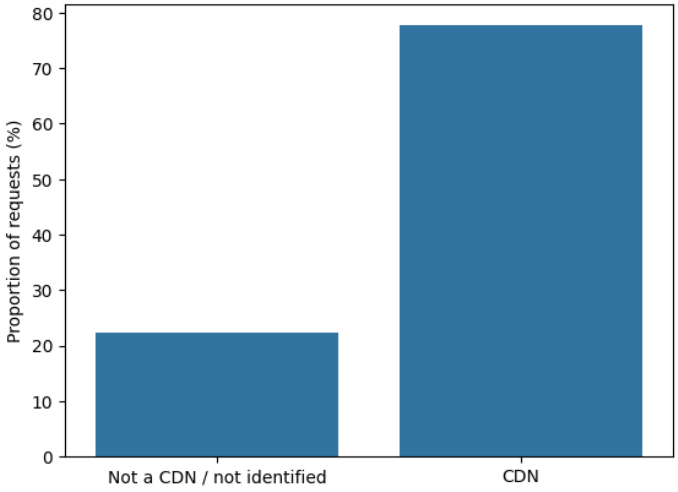

Figure 3: Share of CDNs by request

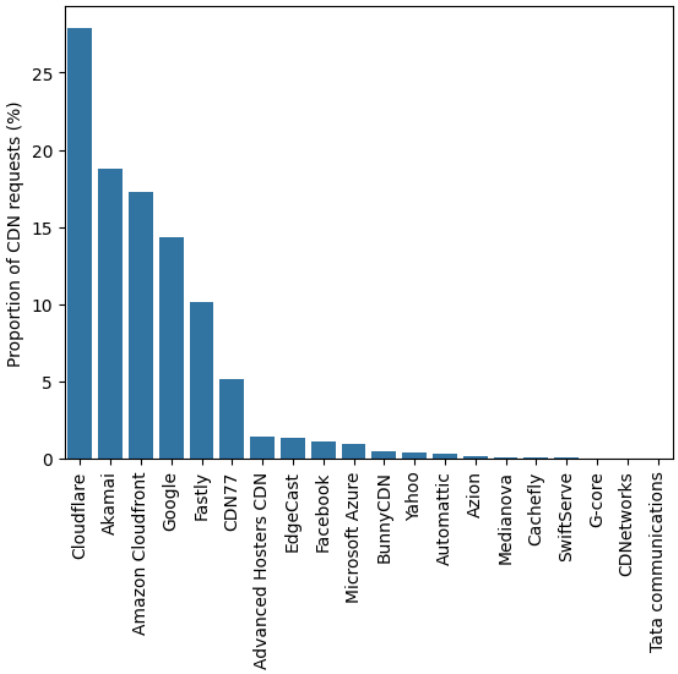

Figure 4: Share of CDN providers by request

4.2 Share of resource types

Another aspect of comparison of CDN providers can be the listing by resource type. This is what can be seen in figure 5 where only the top 10 identified resource types are used for clarity. The requests of the considered resource types are a subset of the recorded requests. Therefore, the decrease is not continuous in the chart. But as values are getting lower towards the right side, the information value and significance of the data decreases. So in figure 6 only the top six CDNs reaching more than 5% share are shown and their values within their bin are normalized to make the distribution of shares within their bin more clear. So a comparison between the providers makes only sense when talking about the shares, not the sizes in the chart. Nevertheless, some special features stand out from the two charts: With Cloudflare (and also CDN77) delivering a low share of HTML files it can be assumed, that these CDNs are mainly used for site assets and that the HTML was delivered from another CDN or from the origin server itself. A setup that would profit from this combination would for example make use of assets that can be cached more easily getting delivered by a CDN and a dynamic or personalized HTML page getting created by the origin server. Google, on the other hand, delivers a bigger share and also is especially used for JavaScript and web fonts delivery but does not seem to be used much for image delivery. The big share of web fonts (woff2) can be explained with the Google Fonts service offering a variety of hosted web fonts for free. The other characteristics probably have to do with the suitability and the products that are based on the CDNs and would require a further analysis.

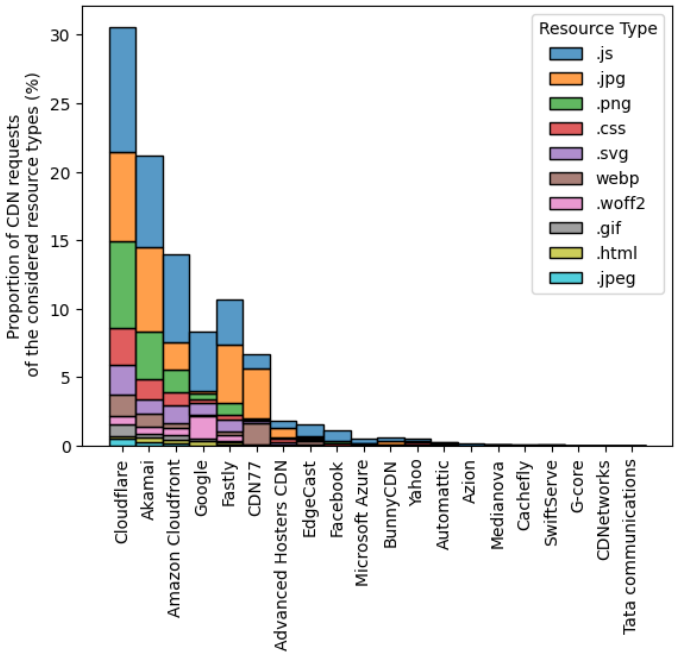

Figure 5: Share of resource types by CDNs

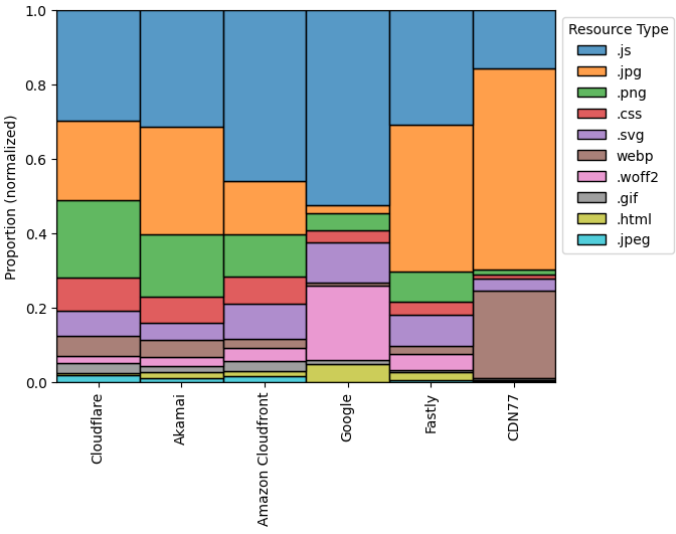

Figure 6: Share of resource types by CDNs (Top 6, normalized)

4.3 Bytes delivered

The number of bytes delivered can also be a factor in determining the popularity of CDNs. With that we get a much more balanced chart for the top 4 CDNs, delivering all more than 250000000 bytes ($\hat{=}$ 250 megabyte) of resources in 7. The amount of bytes was measured using the `Content-Length` response header that sends along the size of the message body [5].

To get an idea of the effect of the two popularity metrics used so far (number of requests, delivered bytes), figure 8 shows the same data as figure 7 but with the metric of number of requests added. In addition, the data is normalized to allow for a comparison of shares that would not be possible with absolute values alone.

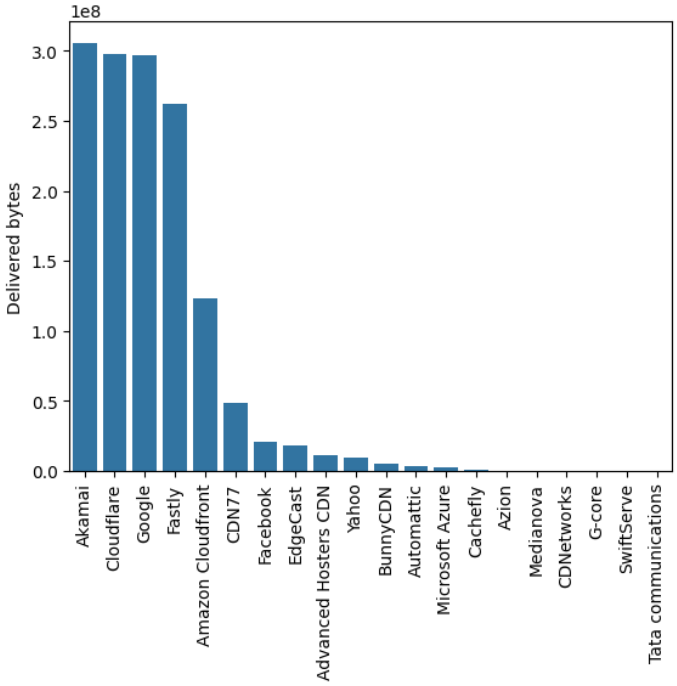

Figure 7: CDN popularity by delivered bytes

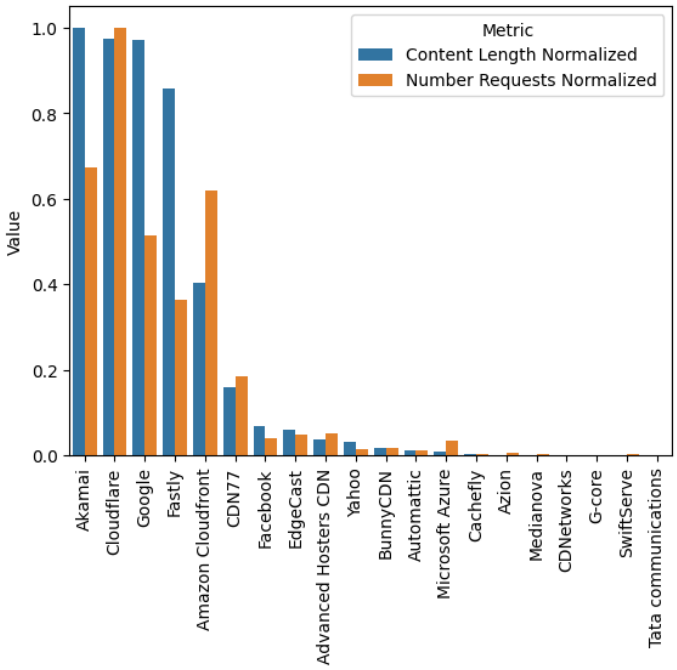

Figure 8: Normalized content length and number of requests per CDN

4.4 Shared resources

The final step of the analysis is a look at shared resources. A distinction is made between shared resources by exact URL and shared resources by name. A shared resource by exact URL is a single, unique file, while the aggregation by resource name includes multiple files that use the same name or naming scheme. But in practice, as figure 9 and 10 show, there is a large overlap, with the largest amounts of matches coming from Google Analytics [4], Google Ad Manager [10] and again Google Fonts. But when only looking at the names, some unique files can be spotted as well. For example, the files `main.js`, `favicon.ico`, `en.json`, and `logo.svg` have rather generic names but typically many pages use files like this for highly customized content and they therefore require a custom delivery.

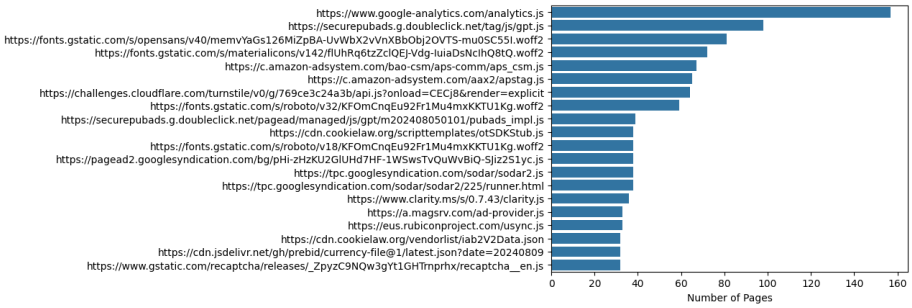

Figure 9: Top 20 full request URLs by number of pages

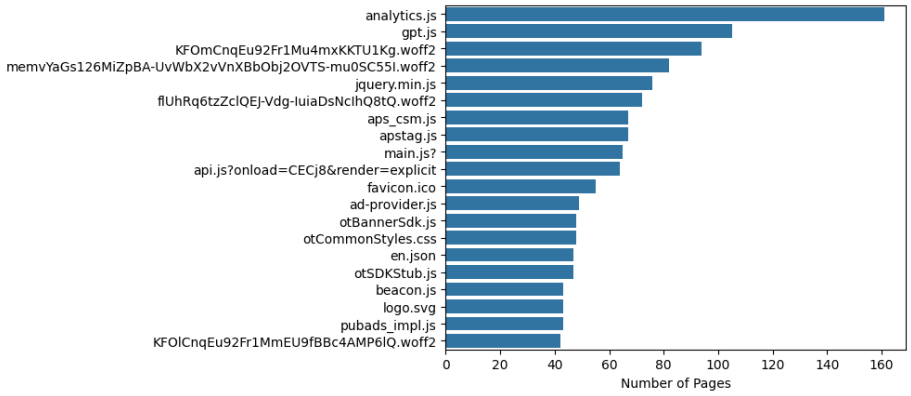

Figure 10: Top 20 resource names by number of pages

5 Conclusion and critical review

In conclusion, this paper has provided an overview of the use of CDNs and CDN providers by the top 1000 websites and identified the "big players" in the field. The results can be seen as confirmation of the continuing trend towards CDNs, with the Web Almanac stating that in 2021 61.1% and in 2022 64% of all requests of the top 1000 websites were served by CDNs [1][2]. So the 77% measured in this paper seems high, but definitely somewhere in the reach, and with that the next Web Almanac release can be expected to go in that direction.

Nevertheless, the results should be treated with caution for several reasons: The results are only an approximation of the real situation and there is no ground truth. Not all hyper-parameters may be set correctly. As a perspective, CDN detection techniques can probably always be improved - there are even companies working on this very thing. This means that not all CDNs may have

been detected, not every CDN may have been detected correctly, some requests may have been misidentified as coming from a CDN, and finally, some sites may rely on multiple CDNs. The latter relies on non-public logic and gives companies that do so the benefit of load balancing and, in particular, resiliency. Also, just looking at a site's home page in most cases introduces a certain bias, as many sites optimize the landing page, showing different content than on other pages or compared to the logged in state. Another bias can be pages that just keep fetching data as long as they are opened such as speed test pages.

Finally, setting up all the parts of the data fetching process as a Python script probably made things more complicated than a combination of a Python script and terminal commands, which might have been simpler and more reliable.

Other exciting areas of research for further analysis could include detailed analysis of individual CDNs and their unique selling points and suitability for different applications.

References

[1] Web Almanac 2021. Cdn. `https://almanac.httparchive.org/en/2021/cdn`.
Last retrieved 24.06.2024.

[2] Web Almanac 2022. Cdn. `https://almanac.httparchive.org/en/2022/cdn`.
Last retrieved 24.06.2024.

[3] Web Almanac 2022. Methodology. `https://almanac.httparchive.org/en/2022/methodology#websites`.
Last retrieved 24.06.2024.

[4] Google Analytics Help. analytics.js. `https://support.google.com/analytics/answer/10268458?hl=en`.
Last retrieved 08.08.2024.

[5] mdn web docs. Content-length. `https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Length`.
Last retrieved 07.08.2024.

[6] CDN Planet. The all new cdn finder. `https://www.cdnplanet.com/blog/all-new-cdn-finder/`.
Last retrieved 24.06.2024.

[7] Chrome UX Report. Crux tools: Crux on bigquery. `https://developer.chrome.com/docs/crux/methodology/tools?hl=en#tool-bigquery`.
Last retrieved 24.06.2024.

[8] Kimberly Ruth, Deepak Kumar, Brandon Wang, Luke Valenta, and Zakir Durumeric. Toppling top lists: evaluating the accuracy of popular web-

site lists. *Proceedings of the 22nd ACM Internet Measurement Conference*, 2022.

[9] Selenium. Webdriver. `https://www.selenium.dev/documentation/webdriver/`.
Last retrieved 24.06.2024.

[10] Google Publisher Tag. Get started with google publisher tag. `https://developers.google.com/publisher-tag/guides/get-started`.
Last retrieved 08.08.2024.

[11] The Verge. Amazon is retiring alexa — no, not that one. `https://www.theverge.com/2021/12/9/22825744/amazon-retiring-alexa-web-ranking-sevice`, 2021.
Last retrieved 24.06.2024.

[12] zakird. crux-top-lists. `https://github.com/zakird/crux-top-lists`, 2024.