Caching / Content Delivery Networks

Fabian Waller, Advisor: Marcel Maltry

I. Introduction

In modern web applications, ensuring predictable end-to-end system quality and performance is crucial. Even minor performance issues can have a significant impact on business, resulting in lost revenue and damage to brand reputation.

However, the Internet's existing architecture was not designed to meet the demanding levels of performance, reliability, and scalability that these applications require. The Internet is made up of thousands of distinct networks, which means that centrally-hosted content must traverse multiple networks to reach end users. As a result, capacity issues arise at peering points where networks exchange traffic. Wide-area Internet communication is vulnerable to various bottlenecks, such as latency, packet loss, network outages, inefficient protocols, and inter-network friction. These limitations within the Internet architecture hinder its ability to deliver static and dynamic web content efficiently and with guaranteed end-to-end system quality. [1]

In addition to these challenges, basic routing protocols such as the Transmission Control Protocol (TCP) were not designed for optimal performance [2]. Route calculations for internet traffic rely primarily on an Autonomous System (AS) [3]. The AS lacks knowledge about topologies, latencies, and real-time congestion in subnetworks.

Content Delivery Networks (CDNs) attempt to mitigate these problems, which are beyond the direct control of a web developer. CDNs emerged in the late 1990s as critical tools for overcoming significant technical obstacles, bridging the gap between the limited capabilities of the Internet infrastructure and the performance requirements of web applications. At the time, bandwidth prices were high, but infrastructure costs were less significant. Most CDN providers therefore aimed to minimise bandwidth requirements by distributing servers with content caches close to end users within the existing Internet architecture, while minimising server loads, client response times and server availability. Since then, the Internet has evolved significantly, with bandwidth prices falling, customer demand for rich media content increasing and server costs rising. As more people consume digital content, the need for enhanced security, additional cloud functionality and support for market metrics and analytics has grown. [4]

A highly distributed network emerges as the most effective architectural solution, especially for interactive and bandwidth-intensive content. CDNs enable companies to achieve very acceptable levels of performance, reliability and cost-effective scalability, but also provide the ability to iterate and ship faster - with much less worry about infrastructure provisioning, capacity planning, architecture for scalability and breaking production code. [1]

In the following, this report will provide a broad overview of CDNs by explaining how they work as a virtual network over the existing Internet infrastructure, what system components are required to work together as a web content cache, and how this differs from other caching methods. Two different cache distribution methods are presented. It also explains how entire applications can be made highly performant by moving application logic closer to the user, and the additional benefits of a recovery-oriented design philosophy.

II. Overview

An origin server is the server that hosts the original version of the content. This can be a web server, an application server, a dedicated storage server or a database server, usually hosted in a larger data centre. Edge servers hold additional copies of this content that are distributed in close proximity to end users. End users only communicate with the edge servers, which are responsible for retrieving content from the origin server if it is not in its cache, significantly reducing the load and bandwidth requirements on the origin server cluster.

As a result, this optimisation has a positive impact on the perceived performance of Web services to users, as it aims to minimise bottlenecks in the middle mile and ensure fast retrieval of cached content when available.

A. Virtual Networks

A CDN, defined as a geographically distributed network of Points of Presence (PoPs) [5] where edge servers are hosted in data centres [6], operates seamlessly over the existing Internet infrastructure as an adaptable virtual network without requiring client software or changes to the underlying networks. [1] CDN nodes are typically deployed on a widely distributed hardware infrastructure comprising tens of thousands of servers around the world, spanning various major data routes [7]. The widespread presence of PoPs around the world ensures that users can access a high-speed server in their proximity, ideally within their local ISP's network, as shown in Fig. 1

B. System Components of a Delivery Network

The CDN components shown in Fig. 2 are designed to work together to deliver content to end users quickly and reliably.

1) Request Handling: When a user initiates a URL request, the **mapping system** translates the domain name into the IP address of an edge server. This system uses data to intelligently select an edge server that is in

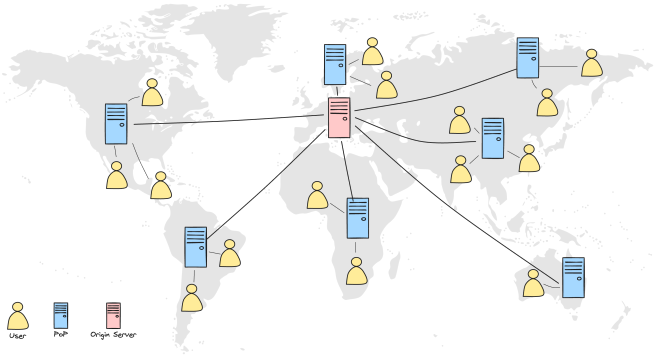

Fig. 1. The edge servers are located in global distributed Points of Presence (PoPs), which are interconnected by an optimised transport system. Either each PoP has its own IP address (DNS-based routing) or all PoPs share the same IP address (Anycast). In either case, users are somehow mapped to the optimal (closest) edge server, which then retrieves the content from the origin server if it is not already in its cache.

optimal proximity to the end user. Content requests are typically algorithmically routed to nodes optimised based on specific objectives such as geographic location, availability (in terms of both current and historical server performance and network congestion), performance, cost considerations, or the dynamic likelihood that the requested content is already in cache. [8] [9]

2) Subsequently, the end user's browser initiates an HTTP request to the obtained **edge server** IP address to retrieve the content. The edge server then checks its cache for the requested content. If the content is cached, the edge server delivers it to the end user.
3) The primary role of the transport system is to move data efficiently and reliably from the origin to the edge servers. In cases where the content is not already cached, the edge server efficiently retrieves it from the origin server via the **transport system** before delivering it to the end user. The transport system is responsible for ensuring a reliable and high performance connection for data and content over the long distance Internet. Communication between CDN servers can be optimised through various techniques such as path optimisation and protocol enhancements. The transport system also accelerates non-cacheable customer content and applications by retrieving content or performing freshness checks from the origin server.

Current status information, control messages and configuration updates are usually available to CDN customers through a **communication and control system**. Often a **data collection and analysis system** systematically collects and processes data, including server and client logs, user data, and network and server information. The **management portal** acts as a configuration management platform and provides analytics based on the collected data, such as audience demographic reports and insights into user interactions with the application, traffic metrics, monitoring, alerting, reporting and billing. [1]

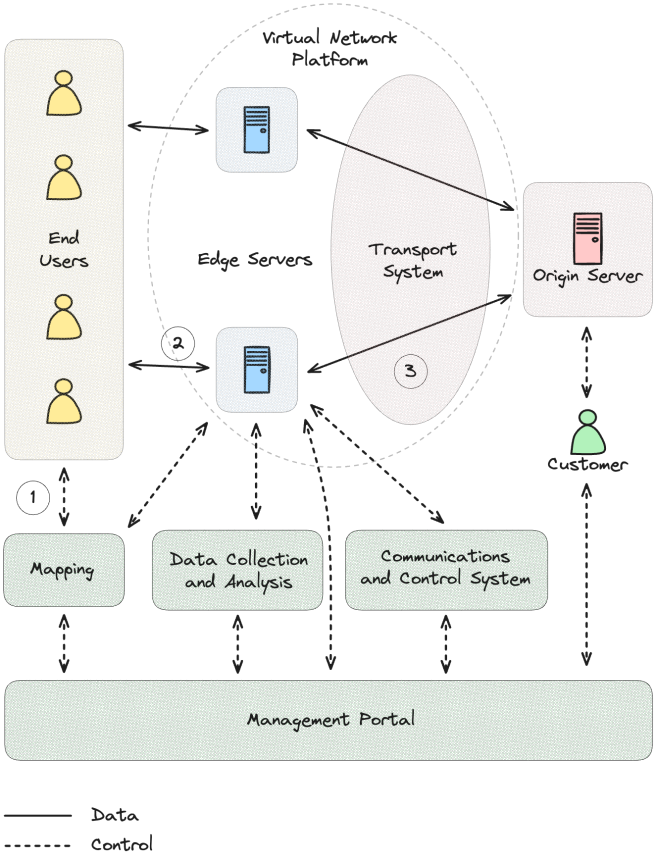

Fig. 2. When a user requests content, the mapping system translates the domain name into the IP address of an optimal edge server. The edge server then checks its cache for the requested content. If the content is cached, the edge server delivers it to the end user. If necessary, an edge server can request content from an origin server (backend web server, application server). The transport system is responsible for ensuring a reliable and high performance connection for data and content over the long distance Internet.

C. Difference from other caching problems

While all caching mechanisms share the overarching goal of reducing latency and improving access times, they are specifically designed for distinct contexts. CDNs operate at the application layer within the technology stack, focusing on delivering content to end users over the Internet. In contrast, database buffers, file system caches and L2 caches operate at the database, operating system and hardware levels respectively. And unlike a local browser cache, which is exclusive to a single user, a CDN is a shared cache accessible to all users of a service. [10] These different caching mechanisms are complementary. For example, a web application benefits from the shared use of a database buffer to speed up dynamic data retrieval, a CDN to deliver web content quickly, and each user's local browser cache to store static content that does not change frequently.

D. Cache Distribution

Web caches store content on servers that have the greatest demand for the requested content. They are filled based on

user requests (pull caching) or based on preloaded content distributed by content servers (push caching).

The tiered distribution model for less frequently accessed content, known as "**pull**", involves the use of a set of well-provisioned "parent" clusters (they have a high degree of connectivity to edge clusters). If an edge cluster does not have the requested content in its cache, it will retrieve the content from its parent cluster instead of the origin server. This approach reduces the load on the origin server, which only needs to maintain connections with a few dozen parent clusters rather than all edge servers. Both origin and parent clusters can use the performance-optimised transport system.

In contrast, the **push** model involves an overlay network, which is particularly useful for live video streaming or edge configuration. The captured and encoded stream is sent to an entry point cluster, and to avoid single points of failure, copies of the stream are sent to additional entry points with automatic failover mechanisms. The live stream transport system then propagates the stream packets from the entry point to a subset of edge servers. Reflectors, an intermediate layer of servers, enable scalable replication of streams to multiple edge clusters, providing alternate paths for improved end-to-end quality through path optimisation. [1]

The distributed nature of the CDN network is key to the effectiveness of the overlay network, ensuring highly optimised long-haul tunnels with endpoints located close to the origin server and end user. This results in optimised communication from origin to end user, making even origin server downloads through the high performance overlay almost as efficient as cached files.

III. High Performance Application Delivery Networks

In addition to static files, entire web applications and other non-cacheable dynamic content benefit from using a CDN in two primary ways.

A. High Performance Overlay Network

First, a CDN takes advantage of the speed of the Internet for long-distance communications by using the CDN transport system as a high-performance overlay network. It traverses the Internet and reaches a CDN machine close to the customer's origin server, usually within the same network or even the same data centre, so that latencies are low.

B. Edge computing

Second, developers can push application logic from the origin server to the edge of the Internet. CDN customers can deploy functions to the edge, where they can be executed based on HTTP requests or custom events. This allows code to run in local data centres, closer to end users, and provides a relatively simple multi-region setup. The ultimate performance boost, reliability and scalability is only achieved when the application itself is distributed to the edge. Deploying and running a request driven application or component on edge servers brings cloud computing to a level where resources are not only allocated on demand, but also close to the end user. But it also brings new challenges, including more complex session management, multi-machine replication, security sandboxing, fault management, distributed load balancing, and resource monitoring and management, as well as advanced requirements for testing and deployment tools. [1]

Not all types of applications can run entirely on the edge, especially those that rely heavily on large transactional databases. However, several applications or parts thereof can benefit, including content aggregation/transformation, static databases, data collection and data validation. Even with real-time database transactions, running front-end components at the edge offers performance benefits by streamlining communications with the origin server and reducing load. For example, the origin server can generate a small dynamic page that references cachable fragments, allowing the final HTML page to be assembled and served at the edge. [1]

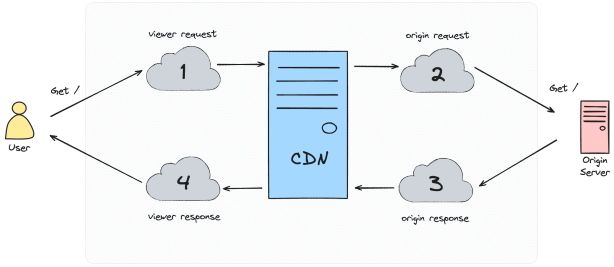

Fig. 3. Edge functions can run at different logical locations, namely viewer request (before caching), origin request (after caching), origin response (before caching origin response) and viewer response (before sending the origin/cached response).

As visualised in Fig. 3, application logic can also run in different logical places for different purposes, such as content paywalls, permanent redirects, image formatting and setting user cookies for analytics. Why these functions are best executed at these locations is explained below.

1) If caching is done on a per-user basis (based on an authentication token), it may result in rarely returning a cached result, as users may only visit the site once a day. Instead, it is recommended to remove the authentication token, extract the subscription level and set a header containing this information before caching the request. This header can then be cached, resulting in a cached response returned to all users with the same subscription level.
2) For permanent redirects, such as during a migration, a URL can be matched and redirected to a new URL. Configuring this after caching ensures that the traffic is routed through the CDN.
3) To handle image formatting, the client first makes a GET request, specifying parameters for image type and size acceptance. After the storage bucket returns the image, which may be larger and in a different format, the edge function transforms it into the correct format. Because this transformation occurs before the response is sent,

the modified image can be cached and served to other users without the need for additional transformations.

4) To track users and improve analytics by matching requests to specific users, a user cookie can be set before the response is sent. It is important that this cookie is not associated with the cache, as caches should be shared between users.

With modern web frameworks, it is also easy to run all site functionality in edge functions only, without using an origin server at all. [11]

IV. Benefits

The recovery-orientated design philosophy outlined below has a number of useful by-products.

A. Design Principles

The entire CDN design is based on the assumption that problems such as machine, cluster, connectivity and network failures will occur at some point in the network. The system is therefore designed for **reliability**. The aim is to achieve close to 100% end-to-end system availability. CDNs ensure full component redundancy to eliminate single points of failure and incorporate multiple levels of fault tolerance. With a large infrastructure, CDNs are able to meet increasing traffic demand and handle traffic spikes. All platform components are highly **scalable**, able to efficiently handle varying levels of traffic, content and customers. On the other hand, the **need for human management is limited**. Because the CDN network is designed with the assumption that components can fail at any time, CDNs are designed as autonomic systems and keep human operating costs low. CDNs have the ability to recover from failures, manage load and capacity shifts, self-tune for optimal performance, and securely deploy software and configuration updates. Humans do not have to worry about most outages or rush to fix them. Moreover, staff can proactively suspend components if they have the slightest concern, as this will not affect the performance of the overall system. Another benefit is the ability to roll out software updates seamlessly. Because the failure of a number of machines or clusters does not affect the overall system, zoned software rollouts can be performed quickly and frequently without disrupting production services. This enables application developers to iterate and deliver their products faster and more frequently. Finally, because CDNs are designed for **performance**, they optimise end-user performance, improve cache hit rates, effectively manage network resource utilisation and promote energy efficiency throughout the system.

B. Additional Benefits

In addition to these improvements, a CDN brings other significant benefits, such as

- **Security**: A CDN leverages its significant network capacity at the edge, playing a key role in providing robust Distributed Denial of Service (DDoS) protection, particularly against large-scale attacks, as shown in Fig. 4. The key strategy is to maintain a network capacity that is significantly greater than that of potential attackers. This not only effectively thwarts DDoS attacks, but also prevents downtime and cost explosions. This approach is particularly effective when the CDN is built on an anycast network, allowing attack traffic to be distributed across a large number of servers. [12]

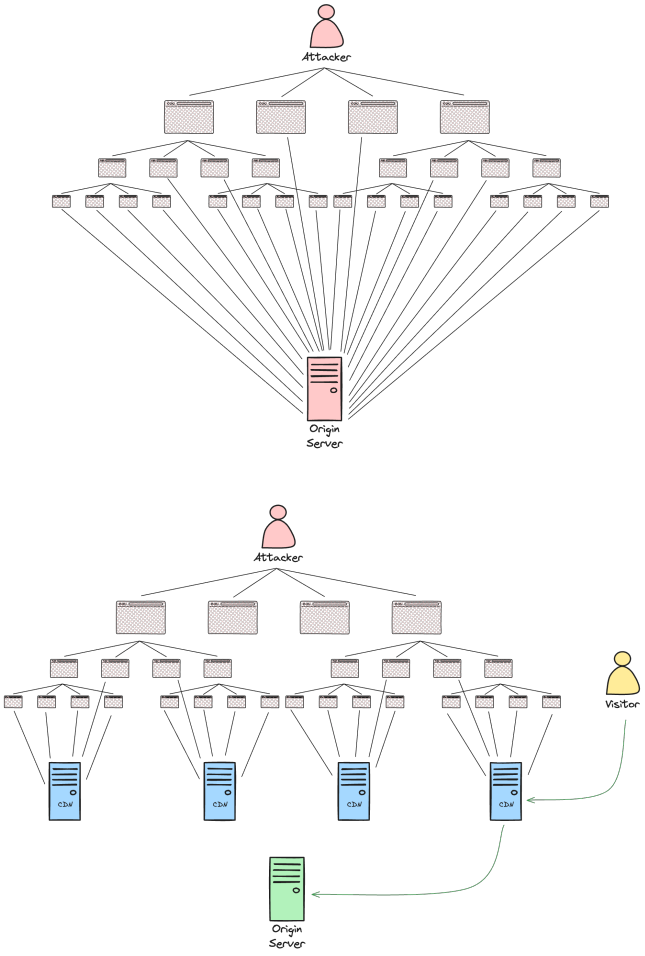

Fig. 4. Web requests are distributed across the different CDN servers. In contrast to a centralised origin server, DDOS attacks do not block the system for other visitors because the load is balanced across many servers.

- **Improved Availability and Reliability**: The inherent design of a CDN is one of high distribution. By having copies of content across many PoPs, a CDN is resilient to multiple hardware failures compared to centralised origin servers, as shown in Fig. 5. The large server distribution acts as a failover mechanism and has a proven ability to maintain uninterrupted services in the face of unpredictable downtime due to machine, cluster or connectivity failures. High availability techniques within edge clusters respond seamlessly to machine failures by starting other machines and timely updating the map used for optimised routing to redirect new requests to accommodate these failures. In the event of whole cluster failures or connectivity issues, the CDN dynamically adjusts cluster allocations and quickly updates the system to redirect

requests to clusters with better performance. The robustness of the CDN platform also extends to connectivity issues, where degraded connections are quickly detected and mitigated through path optimisation technology that finds good alternative paths through intermediate nodes in the CDN network. [12] [11]

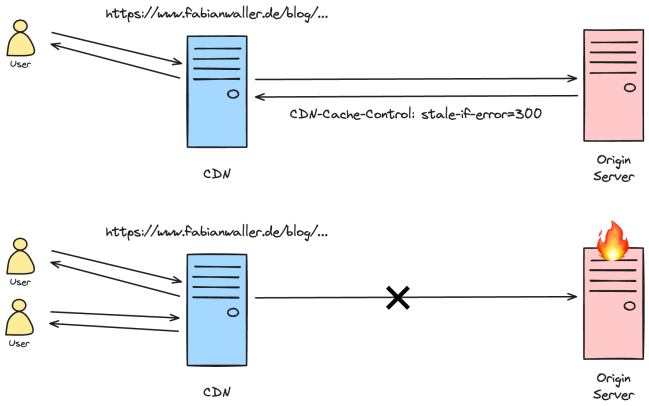

Fig. 5. By having cached copies of content available in many locations, a CDN can withstand many more hardware failures than the origin server alone by potentially serving outdated cached content. There are no more single points of failure.

- **Lower costs** can be a significant financial consideration. CDN egress costs, which refer to the costs associated with data leaving the data centre and reaching the end user, are significantly lower than direct data centre egress. The CDN infrastructure optimises data delivery, reducing the total amount of outgoing data and therefore the financial burden associated with getting data from the data centre to the end user. In addition, using the image transformation edge function mentioned above, or an equivalent built-in functionality, images can be transformed into the optimal format for the end user's browser. This reduces the amount of data that needs to be transferred, further reducing costs.
- Handshakes for encrypted connections take multiple network rounds to establish and are therefore inherently resource intensive. By **terminating the encrypted connection at the edge server**, as visualised in Fig. 6, the latency for users to establish an encrypted connection is significantly reduced. This optimisation is one of the reasons why many modern applications even send dynamic, uncacheable HTTP content via a CDN. [12]

Fig. 6. By terminating the secure connection at the edge, the latency for the user to establish an encrypted connection to the edge server is significantly reduced. The connection to the origin server is kept alive.

The highly distributed nature of the CDN network is key to its effectiveness. This distribution ensures that the endpoints of the optimised long-haul tunnel are located in close proximity to both the origin server and the end user. As a result, most of the communication from the origin to the end user is optimised, with the short hops at either end having extremely low latency due to their short distance. In practice, this optimisation results in good performance over long distances.

- **Flexibility** by providing the ability to integrate with multiple origins. Users can configure routing rules directly within the CDN. This allows customers to define specific rules for content delivery, intelligently routing static file requests to specific storage bucket servers or other appropriate origins. [11]
- Robust **logging and analytics** capabilities are critical to gaining comprehensive insight into system performance. CDNs often facilitate the collection and aggregation of rich data at the edge, providing valuable capabilities for observing traffic patterns, extracting insights and effectively categorising information as mentioned above.
- Modern CDNs often have the ability to go beyond traditional content delivery and actively **transform static content** into more optimised formats. This includes minimising the file size of script bundles, transforming image files into modern formats such as webp, and compressing content. [13] [12]

V. Conclusion

Using a CDN platform provides the desired predictable end-to-end system quality and performance. CDNs were created to reduce the bandwidth required to deliver static web content quickly and reliably, overcoming the inherent limitations of the Internet architecture. They act as a layer on top of the existing Internet infrastructure, operating seamlessly as a virtual network without requiring any software or hardware changes for their customers. End users only communicate with the edge servers, which are then responsible for retrieving content from the origin server if it is not in its cache, significantly reducing the load and bandwidth requirements on the origin server cluster, increasing performance and reducing costs. Even in the unlikely scenario of simultaneous failures, the CDN is highly resilient, recovering quickly and ensuring a consistent and reliable content delivery experience for end users. Overall, the multi-level failover capabilities of a CDN give customers the reliability, availability, security and flexibility they want for their web applications. Today, entire applications can be made highly performant by moving application logic closer to the user.

References

[1] E. Nygren, R. K. Sitaraman, and J. Sun, "The akamai network: A platform for high-performance internet applications," *SIGOPS Oper. Syst. Rev.*, vol. 44, no. 3, p. 2–19, aug 2010. [Online]. Available: https://doi.org/10.1145/1842733.1842736

[2] W. Eddy, "Rfc 9293: Transmission control protocol (tcp)," USA, 2022.

[3] J. Hawkinson and T. Bates, "Rfc1930: Guidelines for creation, selection, and registration of an autonomous system (as)," USA, 1996.

[4] V. Stocker, G. Smaragdakis, W. Lehr, and S. Bauer, "The growing complexity of content delivery networks: Challenges and implications for the internet ecosystem," *Telecommunications Policy*, vol. 41, no. 10, p. 1003–1016, Mar 2017.

[5] "Point of presence (pop)," Nov 2023. [Online]. Available: https://networkencyclopedia.com/point-of-presence-pop/

[6] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl, "Globally distributed content delivery," *IEEE Internet Computing*, vol. 6, no. 5, p. 50–58, Nov 2002.

[7] T. Greene, "What is the internet backbone and how it works," Mar 2020. [Online]. Available: https://www.networkworld.com/article/968484/what-is-the-internet-backbone-and-how-it-works.html

[8] A. Barbir, B. Cain, R. Nair, and O. Spatscheck, "Rfc3568: Known content network (cn) request-routing mechanisms," USA, 2003.

[9] M. Hofmann and L. R. Beaumont, *Content networking: Architecture, Protocols, and Practice*. Morgan Kaufmann, 2005.

[10] Y. Fountis, "How does the browser cache work?" Oct 2023. [Online]. Available: https://pressidium.com/blog/browser-cache-work/

[11] J. Ginnivan. (2023, Jun) Cdns 101: An introduction to content delivery networks - jake ginnivan - ndc oslo 2023. Youtube. Accessed on 03.01.2024. [Online]. Available: https://www.youtube.com/watch?v=djyt_mG3S60&list=PLTD0iXuciM81seuMEtaoRlRgLxo-NpTD2&index=8&t=1597s

[12] ByteByteGo. (2022, Nov) What is a cdn? how does it work? Youtube. Accessed on 03.01.2024. [Online]. Available: https://www.youtube.com/watch?v=RI9np1LWzqw&list=PLTD0iXuciM81seuMEtaoRlRgLxo-NpTD2&index=1

[13] J. A. Sæterås, "Let the content delivery network optimize your images," Apr 2017, accessed on 11.01.2024. [Online]. Available: https://www.smashingmagazine.com/2017/04/content-delivery-network-optimize-images/