In-depth Investigation of CDN: Part 1

Learn push and pull models and dynamic content cache optimization in CDNs.

We'll cover the following

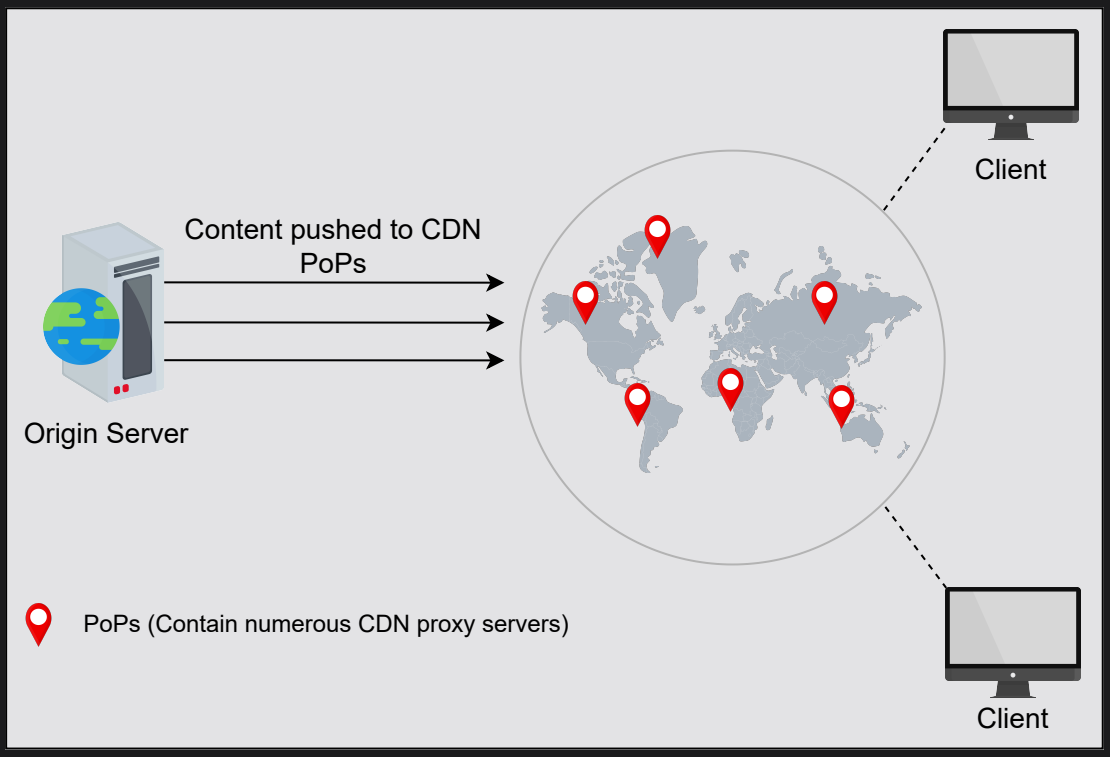
In this lesson, we'll go into the details of certain concepts, such as CDN models and multi-tier/layered CDN architecture, that we mentioned in the previous lessons. We'll also introduce some new concepts, including dynamic content caching optimization and various techniques to discover the nearby proxy servers in CDNs.

Content caching strategies in CDN

Identifying content to cache is important in delivering up-to-date and popular web content. To ensure timely updates, two classifications of CDNs are used t get the content from the origin servers.

Push CDN

Content gets sent automatically to the CDN proxy servers from the origin server in the push CDN model. The content delivery to the CDN proxy servers is the content provider's responsibility. Push CDN is appropriate for static content delivery, where the origin server decides which content to deliver to users using the CDN. The content is pushed to proxy servers in various locations according to the content's popularity. If the content is rapidly changing, the push model might struggle to keep up and will do redundant content pushes.

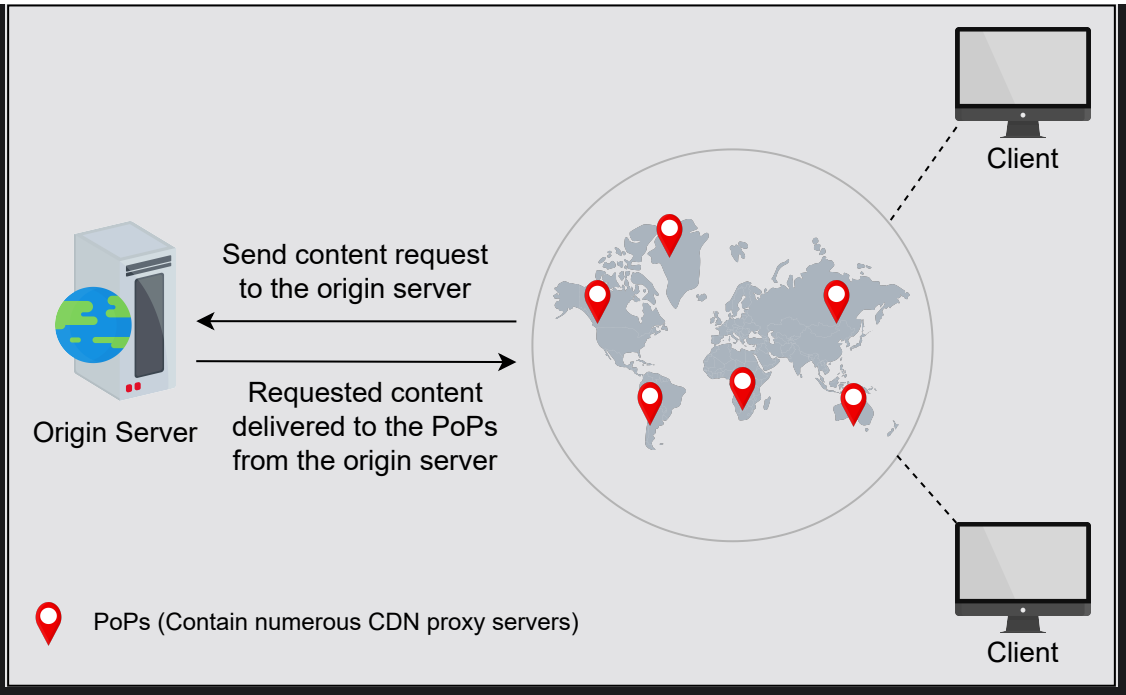


Push content to PoPs

Pull CDN

A CDN pulls the unavailable data from origin servers when requested by a user. The proxy servers keep the files for a specified amount of time and then remove them from the cache if they're no longer requested to balance capacity and cost.

When users request web content in the pull CDN model, the CDN itself is responsible for pulling the requested content from the origin server and servin it to the users. Therefore, this type of CDN is more suited for serving dynamic content.



Content pull from origin server to the CDN PoPs

As stated, the push CDN is mostly used for serving static content. Since static content is served to a wide range of users for longer than dynamic content, the push CDN scheme maintains more replicas than the pull CDN, thus improving availability. On the other hand, the pull CDN is favored for frequently changing content and a high traffic load. Low storage consumption is one of the main benefits of the pull CDN.

Let's say you're managing a website with two distinct content types:

- **News feed:** Frequently updated articles and blog posts with varying popularity.
- **Static assets:** Infrequently updated items, including the company logo, website design elements, and product images.

Choose between a push CDN and a pull CDN for each content type. Justify your choice.

Want to know the correct answer?

Choosing between types of CDN

H1 H2 H3 | B I | | | | | | |

Provide your answer here!

> **Note:** Most content providers use both pull and push CDN caching approaches to get the benefits of both.

Dynamic content caching optimization

Since dynamic content often changes, it's a good idea to cache it optimally. This section deals with the optimization of frequently changing content.

Certain dynamic content creation requires the execution of scripts that can be executed at proxy servers instead of running on the origin server. Dynamic data can be generated using various parameters, which can be beneficial if executed at the proxy servers. For example, we can generate dynamic content based on user location, time of day at a location, third-party APIs specific to a location (for instance, weather API), and so on. So, it's optimal to run the scripts at proxy servers instead of the origin servers.

To reduce the communication between the origin server and proxy servers and storage requirements at proxy servers, it's useful to employ compression techniques as well. For example, Cloudflare uses Railgun to compress dynamic content.

Another popular approach for dynamic data compression is **Edge Side Includes (ESI)** markup language. Usually, a small portion of the web pages changes in a certain time. It means fetching a full web page on each small change contains a lot of redundant data. To resolve this performance penalty, ESI specifies where content was changed so that the rest of the web page content can be cached. It assembles dynamic content at the CDN edge server

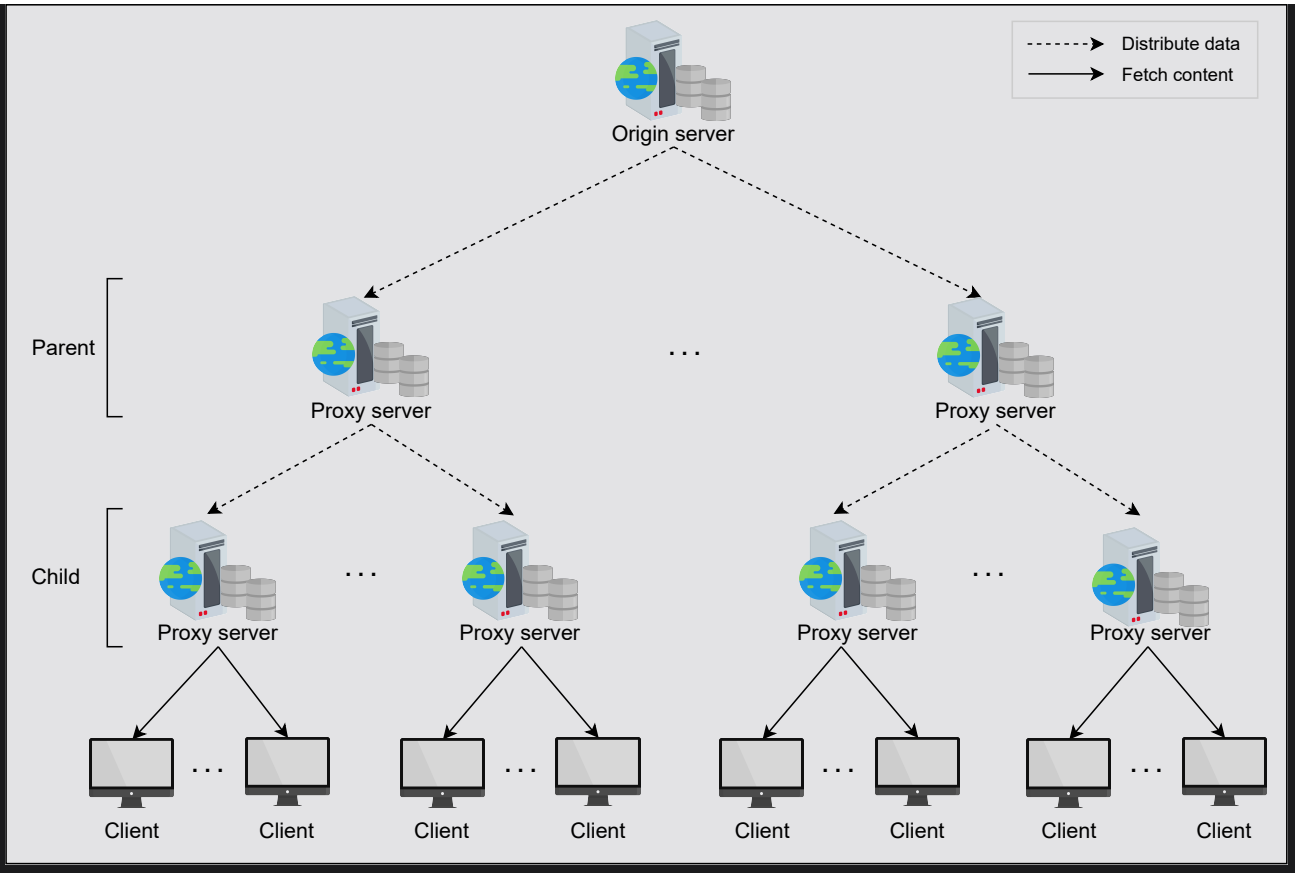
or client browser. ESI isn't standardized yet by the World Wide Web Consortium (W3C), but many CDN providers use it.

> **Note**: **Dynamic Adaptive Streaming over HTTP (DASH)** uses a manifest file with URIs of the video with different resolutions so that the client can fetch whatever is appropriate as per prevailing network and end node conditions. Netflix uses a proprietary DASH version with a Byte-range in the URL for further content request and delivery optimization.

Multi-tier CDN architecture

The content provider sends the content to a large number of clients through a CDN. The task of distributing data to all the CDN proxy servers simultaneously is challenging and burdens the origin server significantly. CDNs follow a tree-like structure to ease the data distribution process for the origin server. The edge proxy servers have some peer servers that belong to the same hierarchy. This set of servers receives data from the parent nodes in the tree, which eventually receive data from the origin servers. The data is copied from the origin server to the proxy servers by following different paths in the tree.

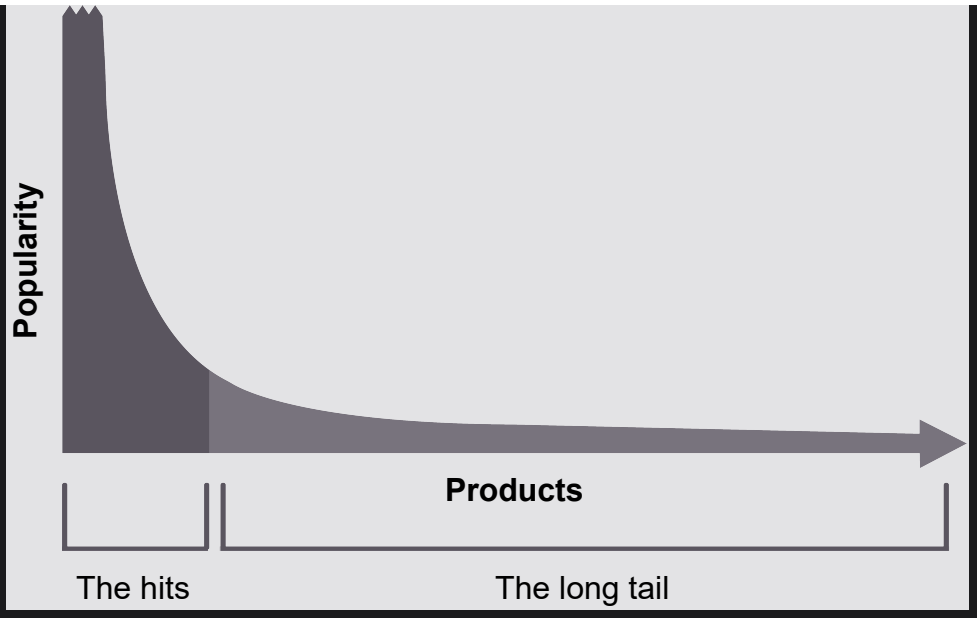
The tree structure for data distribution allows us to scale our system for increasing users by adding more server nodes to the tree. It also reduces the burden on the origin server for data distribution. A CDN typically has one or two tiers of proxy servers (caches). The following illustration shows the two tiers of proxy servers:



Data distribution among CDN proxy servers

Whenever a new proxy server enters the tree of a CDN, it requests the control core, which maintains information on all the proxy servers in the CDN and provides initial content with the configuration data.

Research shows that many contents have long-tail distribution. This means that, at some point, only a handful of content is very popular, and then we have a long tail of less popular content. Here, a **multi-layer cache** might be used to handle long-tail content.



Many kinds of data exhibit the long-tailed phenomenon

Point to Ponder

Question

What happens if a child or parent proxy server fails or if the origin server fails?

Show Answer

Now that we've seen a way to distribute content from the origin server to all the proxy servers of the CDN, we should also educate ourselves on how users can use these proxy servers to get the data more efficiently. We'll discuss how the nearest proxy server is chosen when clients make requests and how the CDN located in the upcoming sections of this lesson.

Find the nearest proxy server to fetch the data

It’s vital for the user to fetch data from the nearest proxy server because the CDN aims to reduce user-perceived latency by bringing the data close to the user. However, the question remains of how users worldwide request data from the nearest proxy server. The goal of this section is to answer that question.

Important factors that affect the proximity of the proxy server

There are two important factors that are relevant to finding the nearest proxy server to the user:

- **Network distance** between the user and the proxy server is crucial. This is a function of the following two things:
  - The first is the length of the network path.
  - The second is the capacity (bandwidth) limits along the network path.

The shortest network path with the highest capacity (bandwidth) is the nearest proxy server to the user in question. This path helps the user download content more quickly.

- **Requests load** refers to the load a proxy server handles at any point in time. If a set of proxy servers are overloaded, the request routing system should forward the request to a location with a lesser load. This action balances out the proxy server load and, consequently, reduces the response latency.

Let’s look at the techniques that can be used to route users to the nearest proxy server.

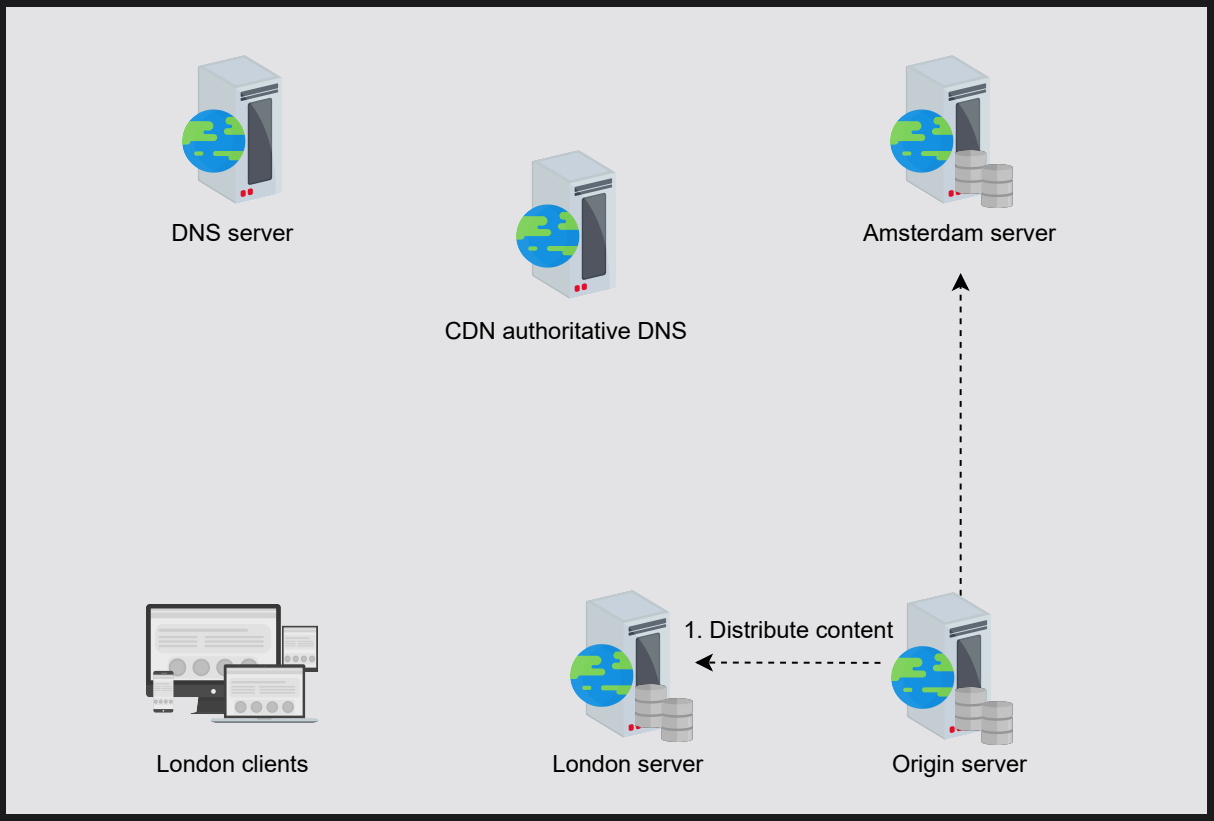
DNS redirection

In a typical *DNS resolution*, we use a DNS system to get an IP against a human-readable name. However, the DNS can also return another URI (instead of an IP) to the client. Such a mechanism is called **DNS redirect**.

Content providers can use DNS redirect to send a client to a specific CDN. As an example, if the client tries to resolve a name that has the word “video” in it,

the authoritative DNS server provides another URL (for example, $cdn.xyz.com$). The client does another DNS resolution, and the CDN's authoritative DNS provides an IP address of an appropriate CDN proxy server to fetch the required content.

Depending on the location of the user, the response of the DNS can be different. Let's see the slides below to understand how DNS redirection works:



The origin server distributes content to the proxy servers

1 of 7

Note: The nearest proxy server doesn't necessarily mean the one that's geographically the closest. It could be, but it's not only the geography that matters. Other factors like network distance, bandwidth, and traffic load already on that route also matter.

There are two steps in the DNS redirection approach:

1. In the first step, it maps the clients to the appropriate network location.
2. In the second step, it distributes the load over the proxy servers in that location to balance the load among the proxy servers (see DNS and Load Balancers building blocks for more details on this).

DNS redirection takes both of these important factors—network distance and requests load—into consideration, and that reduces the latency towards a proxy server.

To shift a client from one machine in a cluster to another, the DNS replies at the second step are given with short TTLs so that the client repeats the resolution after a short while. DNS keeps delivering the content by routing requests to other active servers in case of hardware failures and network congestion. It does so by load balancing the traffic, using intelligent failover, and maintaining servers across many data centers, which achieves good reliability and performance.

Since the load at proxy servers changes over time, the content provider needs to make appropriate changes in the DNS to make the DNS redirection effective. Many CDN providers like Akamai use DNS redirection in their routing system.

Anycast

Anycast is a routing methodology in which all the edge servers located in multiple locations share the same single IP address. It employs the **Border Gateway Protocol (BGP)** to route clients based on the Internet's natural network flow. A CDN provider can use the anycast mechanism so that clients are directed to the nearest proxy servers for content.

Client multiplexing

Client multiplexing involves sending a client a list of candidate servers. The client then chooses one server from the list to send the request to. This approach is inefficient because the client lacks the overall information to choose the most suitable server for their request. This may result in sending requests an already-loaded server and experiencing higher access latency.

HTTP redirection

HTTP redirection is the simplest of all approaches. With this scheme, the client requests content from the origin server. The origin server responds with an HTTP protocol to redirect the user via a URL of the content.

Below is an example of an HTML snippet provided by Facebook. As is highlighted in line 8, the user is redirected to the CDN to download the logo of Facebook:

```
<!--  The code below is taken from Facebook. -->
<div class="fb_content clearfix " id="content" role="main">
 <div>
  <div class="_8esj _95k9 _8esf _8opv _8f3m _8ilg _8icx _8op_ _95ka">
   <div class="_8esk">
    <div class="_8esl">
     <div class="_8ice">
      <img class="fb_logo" src="https://static.xx.fbcdn.net/rsrc.php/y8/
     </div>
     <h2 class="_8eso">Facebook helps you connect and share with the peo
```