Middlewares

Table of Contents

CDN

Definition

A CDN, is a network of edge servers strategically placed across the globe with the purpose of delivering digital content to users as fast as possible.

When a user makes a request it is routed to the nearest CDN edge server, which significantly reduces the latency.

A CDN allows all users, no matter the geographical location, to have fast loading content for an unquestionably improved experience.

Features

Performance

- Content is cached in POPs all around the world bringing content closer to the user.
- The shorter distance will not only reduce latency but also minimize packet loss.

Reliability

- With a CDN, requests will always be routed to the nearest available location.
- If one edge server is not available, requests are automatically sent to the next available edge server.
- This creates automatic redundancy helping to ensure that content always remains available.

Security

- When a CDN is used, the majority of traffic is no longer being served by the origin server, but rather by the CDN edge servers.

- TLS certificates, commonly referred to as SSL certificates, can be implemented on most CDN platforms ensuring all traffic is encrypted
- Morever, many CDNs also have additional security features that can act like a WAF.

Reverse Proxy

- A reverse proxy is a server that takes a client request and forwards it to the backend server.
- A CDN reverse proxy takes this concept a step further by caching responses from the origin server that are on their way back to the client.

Caching

- Caching can took place on CDNs in 2 different ways:

1- Pull Zone

- A Pull Zone will pull files from an existing website without having to upload data manually.
- Most common method to use as many website owners already have an existing web host that is storing all of their files.
- The content gets pulled automatically from the origin server to the POPs and gets delivered to the visitor via the edge servers.

2- Push Zone

- A Push Zone requires data to be upload to the CDN storage cloud manually.
- Typically recommended for distributing larger files, like files larger than 10 MB.
- From here, content is distributed to the CDN POPs in a similar way as a Pull Zone but optimized for larger files.

Forward Proxy

Definition

The forward proxy is served on a network's edge, to regulates outbound traffic.

Features

Forward proxies are typically used internally by large organizations, such as universities and enterprises, to:

- Block employees from visiting certain websites
- Monitor employee online activity
- Block malicious traffic from reaching an origin server
- Improve the user experience by caching external site content

Reverse Proxy

Definition

A reverse proxy accepts a request from a client, forwards it to a server that can fulfill it, and returns the server’s response to the client (on behalf of the webserver), and it can work on different number of webservers from 1 to n.

You can think of the reverse proxy as a website’s "public face." Its address is the one advertised for the website, and it sits at the edge of the site’s network to accept requests from web browsers and mobile apps for the content hosted at the website.

Features

Security

- No Information about backend servers is visibile outside internal network
- They include blacklisting for a certain IP to mitigate DDoS
- They can rate-limit based on the connection of clients
- Reverse proxies are an ideal location to place a WAF to weed out malicious packets.

- From the client's perspective, their requests are resolved via the proxy IP. As a result, your origin server's IP address is masked.

Load Balancing

- Because reverse proxy server are the gateway between users and the application's origin server, they're able to determine where to route individual HTTP sessions, so they cant act as a load balancer.

Caching

- Before returning the backend response to the client, the reverse proxy saves a copy of it locally, when other clients makes the same request, the reverse proxy response back with the cached content, instead of forwarding the request to the backend server.

Compression

- Compressing server responses before returning them to the client (for instance, with gzip) reduces the amount of bandwidth client requires.

SSL Termination

- By decrypting incoming requests and encrypting server responses, the reverse proxy frees up resources on backend servers.

Load Balancer

Definition

A load balancer distributes incoming client requests among a group of servers, in each case returning the response from the selected server to the appropriate client.

Features

Load Distribution

- Most commonly the servers serve the same content, and the load balnacer job is to distribute the workload, in a way that makes the best use of each server's capacity, prevents overload on any server, and results in the fastest possible response to the client.

Reducing errors

- A load balancer can also enhance the user experience by reducing the number of error responses the client sees. It does this by detecting when servers go down, and diverting requests away from them to the other servers in the group.

Session Persistence

- Means sending all requests from a particular client to the same server.

Algorithms

Round Robin

- The simplest method, An inbound request is delegated to the first available server.
- Client requests are distributed to application servers in rotation, first request first sever, second request second server, etc.
- This method is particularly useful when working with servers of equal value.

IP Hash

- The client's IP address simply determines which server receives its request.

Least Connections

- The least connection method directs traffic to whichever server has the least amount of active connections.
- This is helpful during heavy traffic periods, as it helps maintain even distribution among all available servers.

Least Response time

- The least response time method directs traffic to the server with the least amount of active connections and lowest average response time.

Least Bandwidth

- This application load balancer method measures traffic in megabits (Mbps) per second, sending client requests to the server with the least Mbps of traffic.

Cache

Definition

Caching is the process of storing copies of files in a cache, or temporary storage location, so that they can be accessed more quickly.

Web browsers cache HTML files, JavaScript, and images in order to load websites more quickly, while DNS servers cache DNS records for faster lookups and CDN servers cache content to reduce latency.

Different methods

- **Client Side**
- **Server Side**
- **Server Side Utilization of Key/Value stores**

Client Side Cache

- **Browser Cache**

Every time a user loads a webpage, their browser has to download quite a lot of data in order to display that webpage. To shorten page load times, browsers cache most of the content that appears on the webpage, saving a copy of the webpage's content on the device’s hard drive. This way, the next time the user loads the page, most of the content is already stored locally and the page will load much more quickly.

Browsers store these files until their time to live (TTL) expires or until the hard drive cache is full. (TTL is an indication of how long content should be cached.) Users can also clear their browser cache if desired.

Server Side Cache

- **Reverse proxy caches**
- **Load balancer caches**
- **Web application accelerators**

Can be placed in front of application and web servers in order to serve a cached version of the HTTP responses retained from them. These caches are implemented by site administrators and act as intermediaries between the browser and origin server.

Server Side utilization of key/values cache

- **Memcached**
- **Redis**

Unlike reverse proxies which are used to simply cache the HTTP response for a given HTTP request. A key/value object store can be used to cache any web content desired by the application developer, The web content is retrieved typically by means of application code or use of an application framework that can leverage the In-Memory data store. Another benefit to using key/value stores for web caching is they are also commonly used for storing web sessions and other cached content.

CDN Caching

A CDN, caches content (such as images, videos, or webpages) in proxy servers that are located closer to end users than origin servers. (A proxy server is a server that receives requests from clients and passes them along to other servers.)

What types of content does a CDN cache?

- Images: `.png`, `.jpg`, `.svg`, `.gif`, `.tif`
- Style sheets: `.css`
- JavaScript: `.js`

- Video and audio: `.flv (Flash)`, `.mp4 (HTML5 videos)`, `.mov (QuickTime)`, `.wmv (Windows Media)`, `.mp3`, and `.wav`
- Web fonts: `.eot`, `.ttf`, `.otf`, `.cff`, `.afm`, `.lwfn`, `.ffil`, `.fon`, `.pfm`, `.pfb`, `.woff`, `.svg`, `.std`, `.pro`, and `.xsf`
- Other formats: `.pdf`, `.doc`, `.docx`, `.ppt`, `.pptx`, `.xls`, `.xlsx`, `.epub`, `.odt`, `.odp`, `.ods`, `.txt`, `.rtf`, and `.zip`

CDN Cache Hit & Cache Miss

- A **Cache Hit** is when a client device makes a request to the cache for content, and the cache has that content saved.
- A **Cache Hit** means that the content will be able to load much more quickly, since the CDN can immediately deliver it to the end user.
- A **Cache Miss** occurs when the cache does not have the requested content.
- In the case of a **Cache Miss**, a CDN server will pass the request along to the origin server, then cache the content once the origin server responds, so that subsequent requests will result in a cache hit.

If the content is present and delivered from the edge server (CDN) cache the `X-Cache` HTTP header will indicate a HIT. If the content is expired or not present this triggers a MISS.

How long does cached data remain in a CDN server?

- When websites respond to CDN servers with the requested content, they attach the content's **TTL** as well, letting the servers know how long to store it.
- Some CDNs will also purge files from the cache early if the content is not requested for a while
- if a CDN customer manually purges certain content.

Other types of caching

- **DNS Caching** → takes place on DNS servers. The servers store recent DNS lookups in their cache so that they do not have to query nameservers and can instantly reply with the IP address of a domain.
- **Search engines** → may cache webpages that frequently appear in search results in order to answer user queries even if the website they are attempting to access is temporarily down or unable to respond.

Caching Headers

Expires

- Absolute expiry date `Expires: Mon, 13 oct 2020 12:22:00 GMT`
- Old, were used before HTTP/1.1
- Only has an impact on the web browser cache and not the edge server cache.
- Cant be more than a year.
- Clocks have to be in sync.

Pragma

- `Pragma: no-cache` to prevent caching.
- Old, were used before HTTP/1.1
- The same as `Cache-Control: no-cache`

Cache-Control

- Multiple Header value
- New, used on HTTP/1.1

Cache-Control Headers

`Cache-Control: * * *`

Private

- Dedicated to a single user. only cached on the browser.

Public

- Dedicated to multiple users.

- Can be used by intermediary caches (CDNs, proxies, etc.)
- Not necessary to mark this value cause explicit caching information (e.g. max-age) shows that a response is cacheable anyway.

no-store

- Don't cache.

no-cache

- Cache but revalidation needed.
- If a proper `ETag` (validation token) is present as a result, `no-cache` incurs a roundtrip in an effort to validate cached responses.
- Web browsers might cache the assets but they have to check on every request if the assets have changed (304 response if nothing has changed).

max-age

- Content can be cached for the given number of seconds.
- This directive is for client side caching.
- `Cache-Control: max-age=90` indicates that an asset can be reused (remains in the browser cache) for the next 90 seconds.

s-maxage

- Same as `max-age` but "s" stands for shared as in shared cache.
- This directive is explicitly for CDNs among other intermediary caches.
- This directive overrides the `max-age` directive and expires header field when present.

must-revalidate

- Never serve the stale content and must validate the content.
- If `must-revalidate` doesn't exist, it might be possible for the client to be served by the cached content altough the `max-age` expired.

proxy-revalidate

- Same as the `must-revalidate`, however, it only applies to shared caches such as proxies.

no-transform

- Tells any intermediary such as a proxy or cache server to not make any modifications whatsoever to the original asset.
- The `Content-Encoding`, `Content-Range`, and `Content-Type` headers must remain unchanged.

Caching Validators

ETag

- Entity tags (validation token) `Etag: "j82j8322ha7sdh0q2882"`
- Sent by the server in the response.
- A unique identifier assosiated to the resource.
- Strong `ETag` which is: `Etag: "j82j8322ha7sdh0q2882"` → Resources are exactly the same.
- Weak `ETag` which is: `Etag: W/"j82j8322ha7sdh0q2882"` → Resources are not exactly the same but can be considered the same.

Example of ETag

- 90 seconds after the initial fetch of an asset, the browser initiates a new request (the exact same asset).
- The browser looks up the local cache and finds the previously cached response, but cannot use it because it's expired.
- This is the point where the browser requests the full content from the server.
- The problem with it this is that if the resource hasn't changed, there is absolutely no reason for downloading the same asset that is already in the CDN cache.
- Validation tokens `ETag` are solving this problem.

- The edge server creates and returns arbitrary tokens, that are stored in the `ETag` header field
- Which are typically a hash or other fingerprints of content of existing files.
- Clients don't need to know how the tokens are generated but need to send them to the server on subsequent requests.
- If the tokens are the same then resources haven't changed thus downloads can be skipped.
- The web browser provides the `ETag` token automatically within the `If-None-Match` HTTP request header.
- The server then checks tokens against current assets in the cache.
- A `304 Not Modified` response will tell the browser if an asset in the cache hasn't been changed and therefore allowing a renewal for another 90 seconds.
- It's important to note that these assets don't need to be downloaded again which saves bandwidth and time

If-None-Match

- The web browser provides the `ETag` token automatically within the `If-None-Match` HTTP request header.
- Server compares `If-None-Match` against the `ETag`

Last-Modified

- Indicates the time a document last changed which is the most common validator. `Last-Modified: Wed, 15 Sep 2020 12:30:16 GMT`

If-Modified-Since

- When a cache stores an asset including a `Last-Modified` header, it can utilize it to query the server if that representation has changed over time.
- This can be done using an `If-Modified-Since` request header field.
- Client make the conditional request to see if the content is `304 not modified`
- Server: `Last-Modified: Wed, 15 Sep 2020 12:30:16 GMT` Client: `If-modified-since: Wed, 15 Sep 2020 12:30:16 GMT`

According to RFC2616, An HTTP/1.1 origin server should send both, the `ETag` and the `Last-Modified value`

- Example Response

```
HTTP/1.1 200 OK
Server: keycdn-engine
Date: Mon, 27 Apr 2015 18:54:37 GMT
Content-Type: text/css
Content-Length: 44660
Connection: keep-alive
Vary: Accept-Encoding
Last-Modified: Mon, 08 Dec 2014 19:23:51 GMT
ETag: "5485fac7-ae74"
Cache-Control: max-age=533280
Expires: Sun, 03 May 2015 23:02:37 GMT
X-Cache: HIT
X-Edge-Location: defr
Access-Control-Allow-Origin: *
Accept-Ranges: bytes
```

- Client will send both the parameters in the validation headers using `If-None-Match` and `If-Modified-Since`
- Server checks both values to respond with either `304 not modified` or a new fresh one.

References

- Keycdn.com
- Cloudflare.com
- Developer.mozilla.org
- Nginx.com

By HolyBugx