What's a Cache?

They are cache money for improving performance

Justin
Aug 16, 2022

Front matter: Technically referrals

First: **an official Technically referral program!**

Finally, we've got referrals set up. Get your friends and enemies to sign up for Technically and earn exclusive prizes, like stickers (sick ones tbh), a sweatshirt, or a free year subscription. It's easy as hell: just generate your unique referral link with your email address and share it far and wide.

If you run into any issues, let me know. Now, back to our regular scheduled programming.

The TL;DR

Caching is the process of storing things (webpages, images, etc.) closer to whoever is requesting them so that those things load faster.

- When you load a web page or an app, you're **requesting files from a server** – and that server can be physically far away
- Developers will **cache content on servers closer to you** so that those web pages and apps will load faster
- Because web page and apps change all the time, you can set a **timer** on cached content to **force it to expire**
- Specialized server networks called **CDNs** let you cache content easily across the globe

Almost every website you use is being cached in some format or another, and it's an important part of how modern web apps are built (also, the kind of thing they don't teach you in CS degrees).

Saving private latency

Nobody likes a slow web page (well, nobody that I know). But if you think about all of the stuff that needs to happen between you typing gmail.com into your browser and you actually seeing your email, it's a complete miracle that it happens so fast – almost always in less than a second. Let's break it out:

- Client makes a request to the server (i.e. you type in gmail.com)
- Server figures out what to send back to the client (query databases, etc.)
- Server sends back assets (webpage, images, etc.) to client
- Client's browser renders the page and assets

That's *already* a lot, and we haven't even covered protocol procedures like TCP handshakes. Each one of these steps takes time (usually a few hundred milliseconds), and when servers are far away, those steps can really add up. If you think about it, you can break down these "speed problems" into two core issues, which both conveniently start with the letter L:

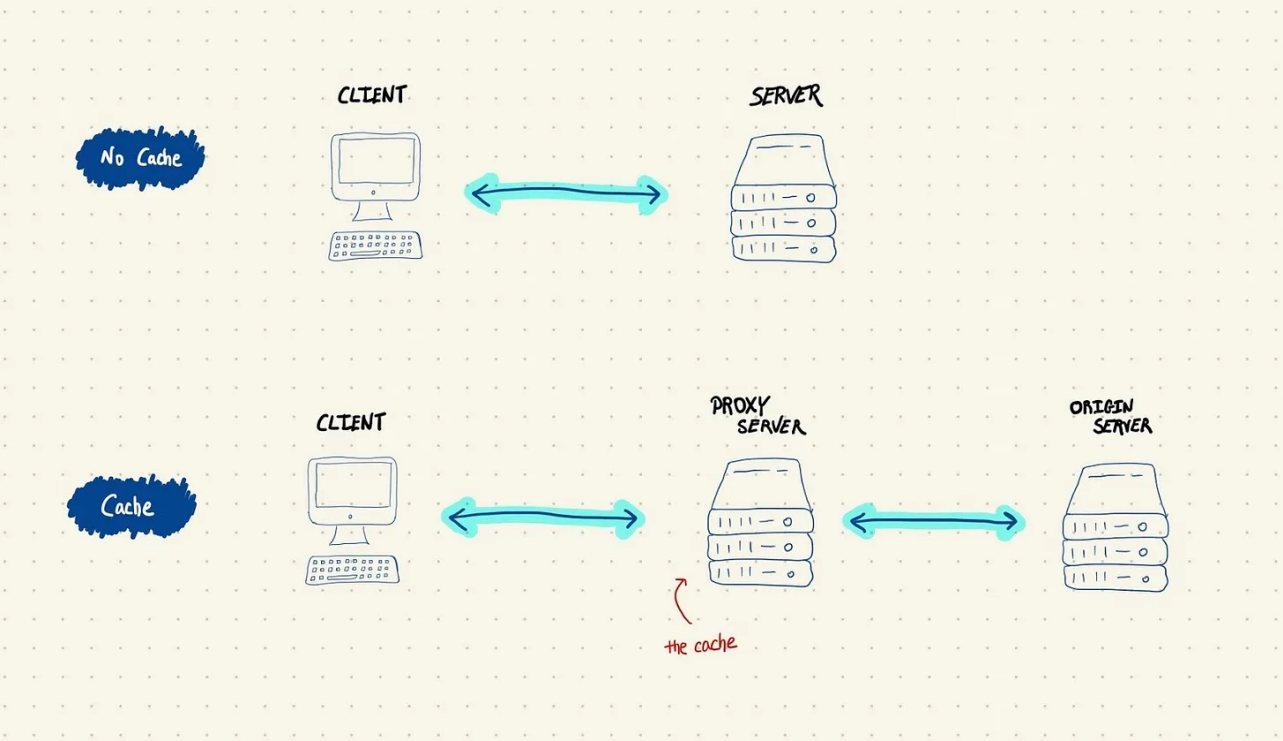
1. **Logic** – there are a steps along the way to getting a web page loaded
2. **Location** – servers can be very far away, and data takes time to travel

Caching helps solve these problems by storing readily available files (pre-made web pages, images, etc.) on special servers that are closer to an app's end users. The shorter physical distance that the content has to travel speeds up load times, and the pre-built files save the original web server from having to deal with request logic again.

> 😕 **Undefined Terms** 😕
>
> An **origin server** is the core web server that an app or site uses to respond to requests and build web pages. A **proxy server** is an intermediate server that stands between the user and the origin server and takes care of things like caching.
>
> 😕 **Undefined Terms** 😕

Logistically, you cache files by either setting up a special server on your own (less common), or using a service like AWS Cloudfront or Cloudflare. We'll talk

more about these below.



One thing to keep in mind: the Nielsen Norman Group's widely used standard says that response times of more than 1 second begin to interrupt the user's flow of thought and engender a negative user experience. For that reason, companies put a lot of effort into making their sites and apps as fast as possible – and that's the context for how caching helps.

Caching 101: RTT, TTL, and assorted acronyms

You can classify caching under the "Web Technologies and Protocols That Are Very Scary and Use a Lot of Acronyms" folder. Let's walk through the anatomy of a cache and define some of the terms that you'll come across.

RTT (Round Trip Time) – the total amount of time it takes from the moment that a user makes a request (you type a URL into your browser, or click a link) until the result gets returned to the requestor. The goal of a cache is to lower this.

TTL (Time to Live) – caches have one big problem: they're static, which means that if you change your app (a new feature, a bug fix, etc.) the cache is still going to have the old version. To get around this, file caches usually carry a TTL

(seconds, minutes, hours) - you can define how long you want your cached content to last for before it needs to get refreshed.

Hits and misses – developers often use the term "hit" to refer to when you make a request to the server and the content is cached (success!). A miss is when the content isn't cached (or needs to be refreshed), so your request needs to go all the way to the origin server to get fulfilled.

> 🚨 **Confusion Alert** 🚨
>
> If there's a cache miss, that doesn't mean you don't get your web page - it just means that the proxy server needs to forward your request on to the original server, get the results, send them to you, and then cache them for future use.
>
> 🚨 **Confusion Alert** 🚨

CDN (Content Delivery Network) – building and operating your own servers just to implement a cache is kind of annoying, and there are options for outsourcing that. A CDN is a network of servers (usually global) that you can tap into and cache your content on: the classic example is Cloudflare, but AWS, Google, and Azure have their own too.

Compression and minification – once you commit to using a cache, you can mess around with your files to make them easier to send across the web. If you've ever unzipped a file, you're already familiar with compression, and a bunch of cache servers have compression protocols enabled by default. Developers will also minify their code by removing comments, spaces, new lines, and anything needed for human readability. Both of these make files smaller, which speeds up travel and lowers that key metric: RTT.

Types of caching: server and browser

The typical cache setup involves figuring out how to properly keep your files on a server, dealing with expiration via TTL, and all of that jazz. But a server or CDN isn't the only place where caching happens; there's plenty of data that's cached right on your laptop, phone, or computer – in your browser. Browsers *also* cache things, especially if you're visiting a site a lot.

Your browser cache behaves like a specialized cache server, but saves *even more time* because your requests don't have to leave your browser at all. For example: if you're loading the New York Times homepage, basic parts of the layout like the boilerplate HTML or the CSS might get cached. That way, when you reload the page, you won't need to fetch *all* of those files again.

Browser caches help explain why reloading pages can be faster than loading pages the first time, and why you'll sometimes have a "stale" page that you keep reloading but doesn't show the new thing that you're looking for. You can clear your browser cache whenever you want.

Terms and concepts covered

`Cache`

`Origin server`

`Proxy server`

`RTT`

`TTL`

`Hits and misses`

`Compression`

```
Minification
```

Further reading

- Caching your assets and using a CDN can have really significant impacts (think: 2x speed) on performance. Cloudflare does a great job of running through the numbers here
- The benefit of a CDN is twofold: caching improves performance in general, and you can also take advantage of data centers available globally to be closer to your users

---

6 Comments

Write a comment...

Ramesh Aug 18, 2022 ♥ Liked by Justin

Very nice, and its more deepdive after reading the cloudflare link provided on bottom of your article.

♡ 2 Reply Gift a subscription Collapse •••

1 reply

Hemanth Chandra Aug 17, 2022 ♥ Liked by Justin

great article !!

♡ 2 Reply Gift a subscription Collapse •••

4 more comments...
