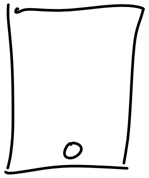
Thank you for reading my zine!

Made with iPad

(with not so great handwriting)



Thanks to:

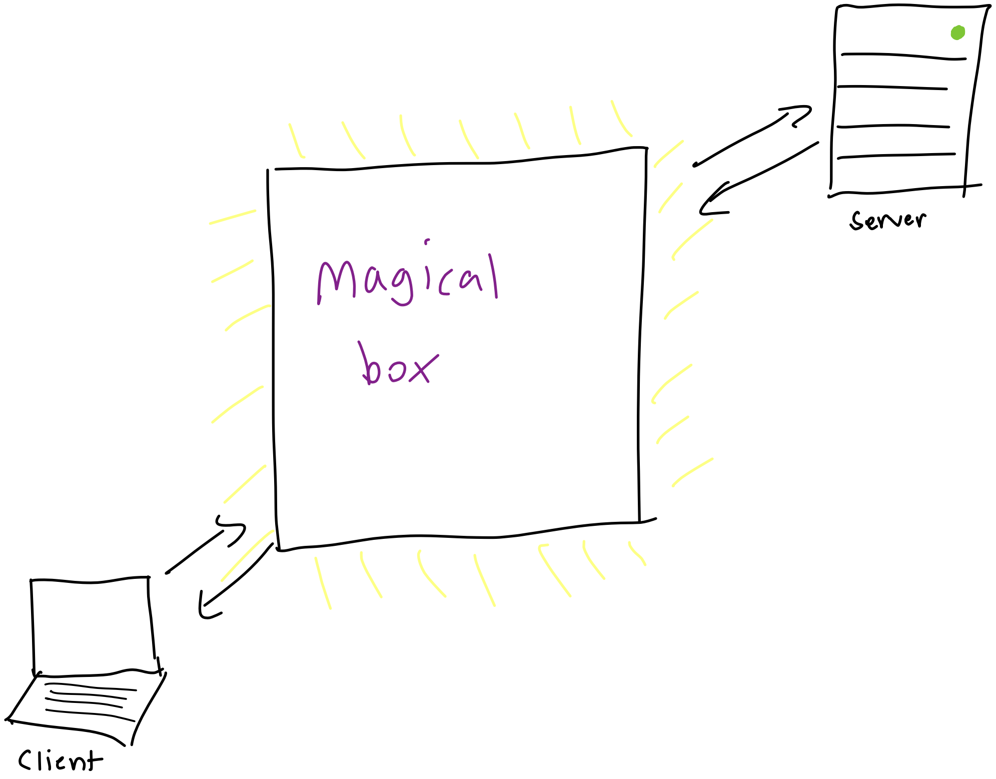
Lindsey Kuper
Julia Evans
Cynthia Taylor

nahtnam.com
light.js.org

What Makes the Web Fast?

by Manthan Mallikarjun

(apologies in advance for the poor typography and art)



Delivering content at 360,000,000 miles per hour

Look inside to learn more about:

CDNs, Anycast, Fingerprinting, Serverless, and more!

Table of Contents

Citations, Sources, and More

[1]: What is a CDN? (cloudflare.com/learning/cdn/what-is-a-cdn/)

What is Anycast? (cloudflare.com/learning/cdn/glossary/anycast-network/)

What is an Origin Server? (cloudflare.com/learning/cdn/glossary/origin-server/)

What is an Edge server? (cloudflare.com/learning/cdn/glossary/edge-server/)

CDN Learning Center (akamai.com/us/en/cdn)

Serverless Architecture (twilio.com/docs/glossary/what-is serverless-architecture/)

What is serverless Architecture? What are its pros and Cons? by Faizan Bashir

The asset Pipeline (guides.rubyonrails.org/asset_pipeline.html) ← for fingerprinting

light (light.js.org)

My brain - a lot of what I have here is stuff I learnt over years of web development

Learn More

The best place to learn more about CDNs is on CloudFlares website. They are highly documented and well written

cloudflare.com/learning

If you know what CDNs are and are interested in highly technical writings about distributed systems, programming languages, and performance, I would highly suggest the CloudFlare blog.

blog.cloudflare.com

You will have to search around to find the good ones.

Glossary

Origin Server

The server that contains the original copy of the files. This is typically programmed and ran by the developer/company

Host

The server itself or the Company that provides the server. Usually this is confusing but if it is in terms of a request, it usually means the server, but if the context is hosting, then the host can mean Amazon, Google, etc.

Edge Server

The servers that are distributed across the world, usually by CDN providers. They usually store a copy of the files

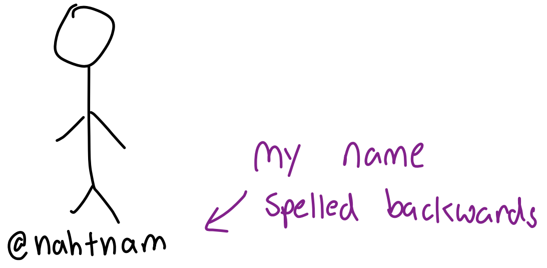
Fact on first page: (assumes page can be served in 100ms)

$$\left(10{,}000 \text{ miles}/100\text{ms}\right) \cdot 10 = 100{,}000 \text{ mps} \cdot 60\text{s}/\text{min} \cdot 60 \text{ min}/\text{hr}$$

$$= \boxed{360{,}000{,}000 \text{ mph}}$$

Who am I?

Hello, my name is Manthan. This is me:



I am a full stack developer

I make websites for fun

I use services such as CloudFlare, Netlify, and Akamai, which are popular CDN providers.

My websites load in under 1s and my APIs in under 100ms. But I take all of this for granted, so I wanted to research what is really happening under the hood, allowing my website to respond to someone 10,000 miles away in under 100ms (360,000,000 mph) at half the speed of light!

Website: nahtnam.com
Blog: blog.nahtnam.com
GitHub: github.com/nahtnam

Yeah, I have my username almost everywhere!

Introduction

If all of the optimizations that are discussed in this zine were to disappear, the web as a whole would slow down by at least a magnitude of 10x.

Why is that? It mostly comes down to location. Just like in real life, packages that are shipped from your country arrive faster than packages that are shipped from another country.

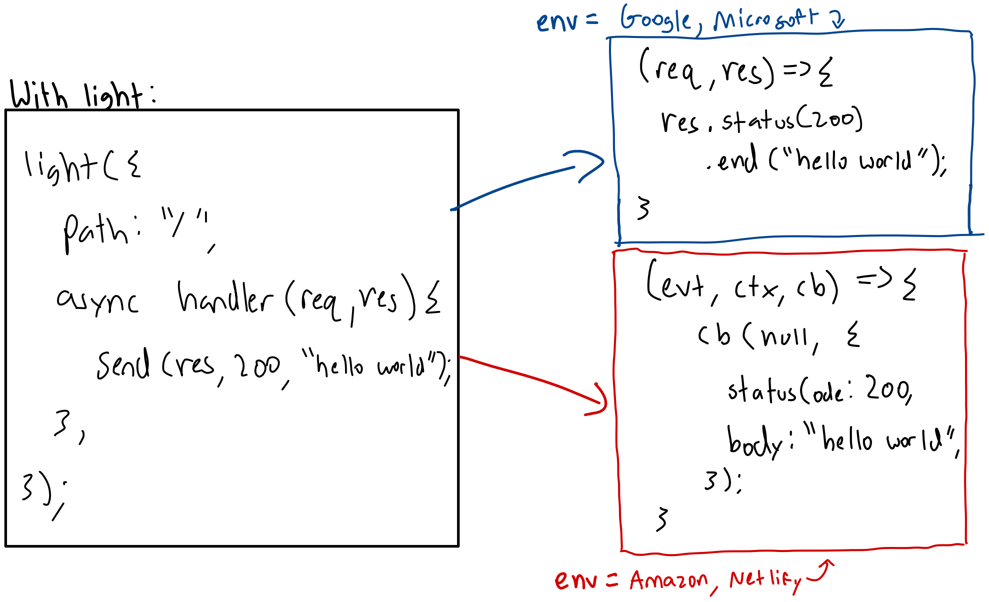
If Google or Facebook only had one server in the US and someone tried to make a request from Asia or Europe, the data would have to travel 1000s of miles across the Atlantic Ocean.

THE CLOSER SOMETHING IS, THE FASTER IT RESPONDS!

Shameless Plug - light (sorry, this is slightly off-topic)

Well there is a solution for this! I have been working on a framework called light which aims to normalize the different serverless environment into one. It also implements features such as HMR (hot-module reload) which updates the server without having to restart it, which is something no popular node framework does.

So with light, you are given standard node Request and Response objects to work with which transform in different environments like so:



With light:

```
light({
  path: "/",
  async handler(req, res) {
    send(res, 200, "hello world");
  },
});
```

```
(req, res) => {
  res.status(200)
     .end("hello world");
}
```

```
(evt, ctx, cb) => {
  cb(null, {
    statusCode: 200,
    body: "hello world",
  });
}
```

You can write once, and deploy anywhere...

Serverless

What if... We use the same tricks we described for the CDNs but with servers instead? Well that is what companies like Amazon, Google, Microsoft, CloudFlare, and Netlify are trying to do.

You send a service a function written in a way they specify. The host will then send your function to all of their edge servers.

Google, Microsoft

```
(req, res) => {
    res.end("hello world");
}
```

```
(evt, ctx, cb) => {
    cb(null, {
        statusCode: 200,
        body: "hello world",
    });
}
```

Amazon, Netlify

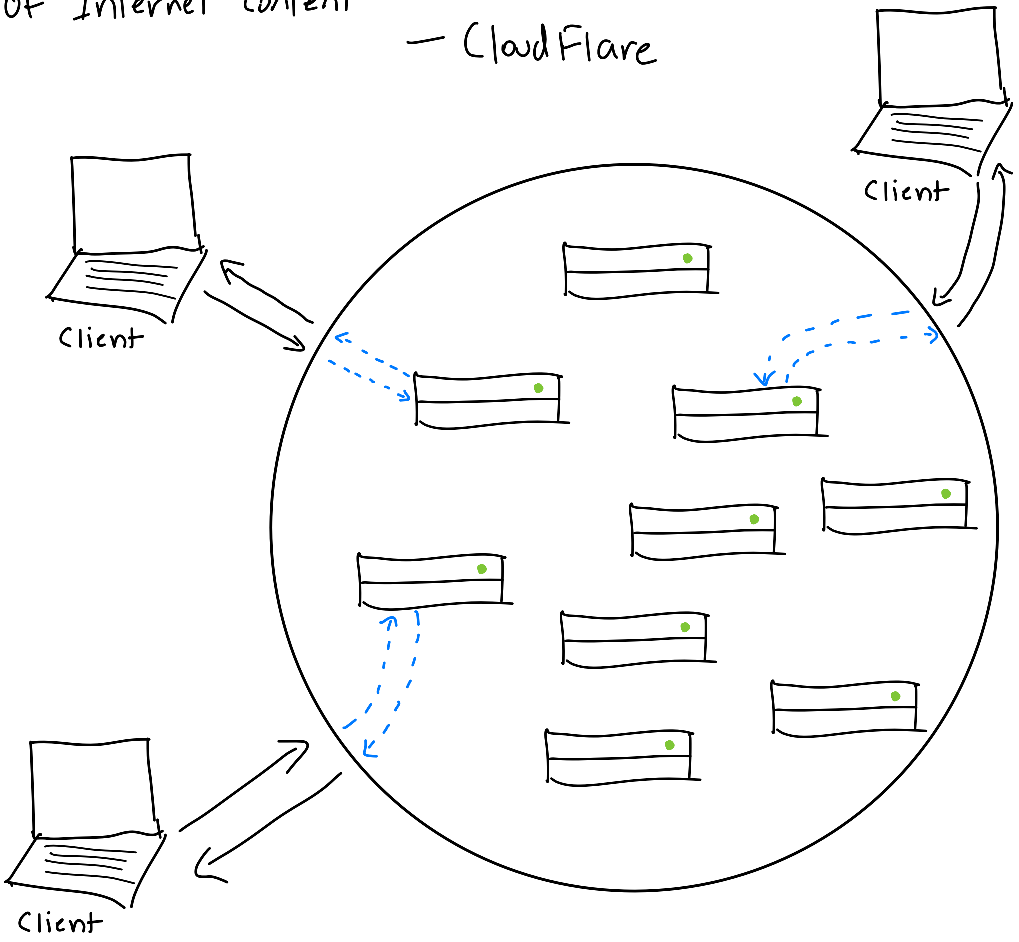
Now, when a client asks for something dynamic, they only have to go to the nearest edge server to compute your data. This not only makes requests faster, it also removes the need for scaling and mitigates DDOS attacks.

Problem: Well what if you write your functions for Amazon, but decide to move to Google? Well then you are screwed... or are you?

What is a CDN?

"A content delivery network (CDN) refers to a geographical distributed group of servers which work together to provide fast delivery of Internet content" [1]

— CloudFlare

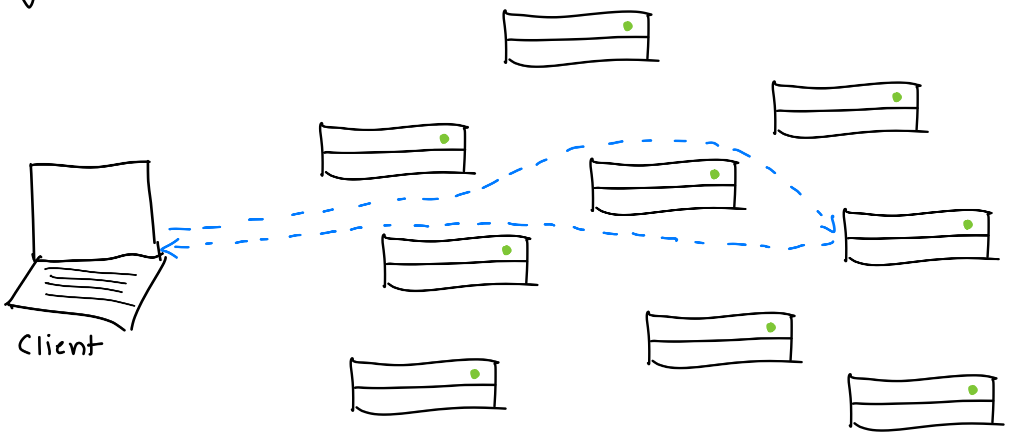


CDNs are responsible for caching static JS/CSS/PNG/ETC files in edge servers* which are physically closer to the client

* See glossary for definition

Unicast and why CDNs don't use it

Unicast is the typical/default way the internet works. Each server is assigned an unique IP address, and when a client makes a request, it will allways go to that specific server and wait for a response. This is great because you can always find the computer you need. But what if it is across the world? How do you find the closest version of that server? What if the server goes down?



There are many questions that remain unanswered with unicast, but that doesn't mean it is bad. Unicast allows for a much simpler system which is why many smaller websites stick with it. However, when handling billions of requests it simply will not work.

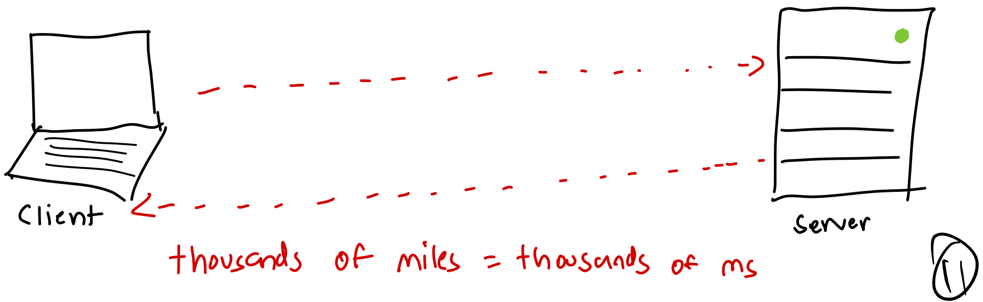
Servers/API

Now that we have talked about static files such as (style.css, script.js, etc.), what about dynamic content that is handled by servers?

Well typically, it is up to the developer or company to buy and redirect people to servers around the world. This is both expensive and cumbersome which is why many just have one server and ignore complaints about how the website is slow

Recently there have been new softwares such as Kubernetes which help companies set up clusters and manage nodes easily. However, these softwares are so complex, companies will still have to hire dev-ops engineers just to set it up.

If only there were a better solution...



Fingerprinting (2)

Instead, there is a simpler solution. What if we were to hash the file and include it in the file name? So the red file would be Style-4ca35.css and the blue would be style-a31b7.css.

Now when you visit the website for the first time, it will load and store style-4ca35.css on the edge server. Once the new file is up, and you load the index.html, it will request for style-a31b7.css. The edge server will realize it is missing that file, it will fetch and store that file.

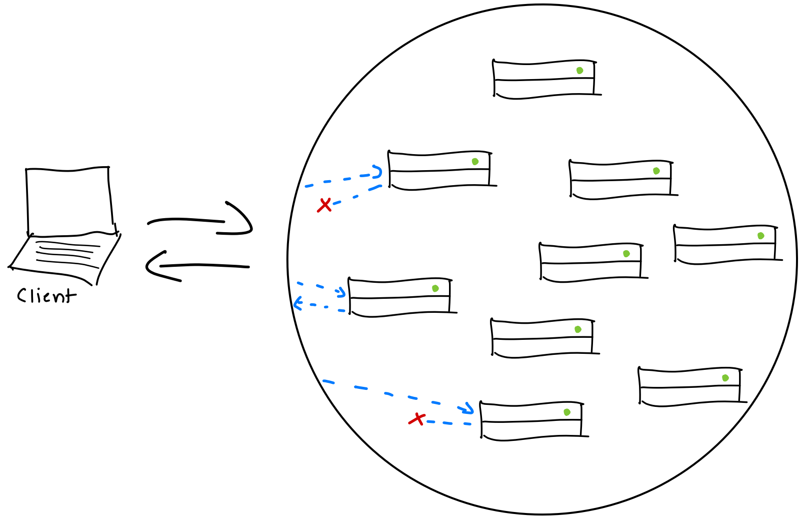
This method will ensure that the CDN will automagically update AND that the correct files are loaded together with the specified index.html

*Note: This is usually done during the build phase with tools such as Webpack or gulp. Ruby on Rails is also a popular framework that fingerprints.

Anycast and why it is better

CDNs primary performance benefits come from the use of anycast.

With CDNs, the client is still given a single IP address. However, when this IP address is requested, it will broadcast the request to multiple servers around the world. The closest server will respond the quickest.



Additionally, anycast also helps with Distributed Denial of Service (DDoS) attacks by spreading the load over a larger number of servers rather than a single server (unicast)

Additional Benefits

Faster internet: When you request a page you will first connect to the CDN. If the CDN does not have the file, it will fetch it from the origin server for you using its datacenter internet.

Cheaper Servers: Since the data is pushed off to the CDN, very few requests will hit your origin server.

Additional Security: CDNs also have the ability to alter your files to make them more secure. So they can <script> tags with additional tags that will verify the checksum to make sure that the file has not been tampered with.

Fingerprinting (1)

There is one big issue with CDNs. How do we update our sites? CDNs will periodically check to make sure that the cache is updated but that is usually not enough.

Thankfully, there is a trick that developers can use to ensure cache is updated instantly. Say you have a style.css file which is the red box. We can easily upload and deploy this to the CDN. Then lets make a change.

```
body {
  background: red;
}
```

```
body {
  background: blue;
}
```

Once we change it to the blue box, we can deploy it again. Chances are that the website will still be red for another half an hour or so. How do we fix this?

Some CDNs have a "purge cache" button but usually it is a hassle to do after every single deployment.