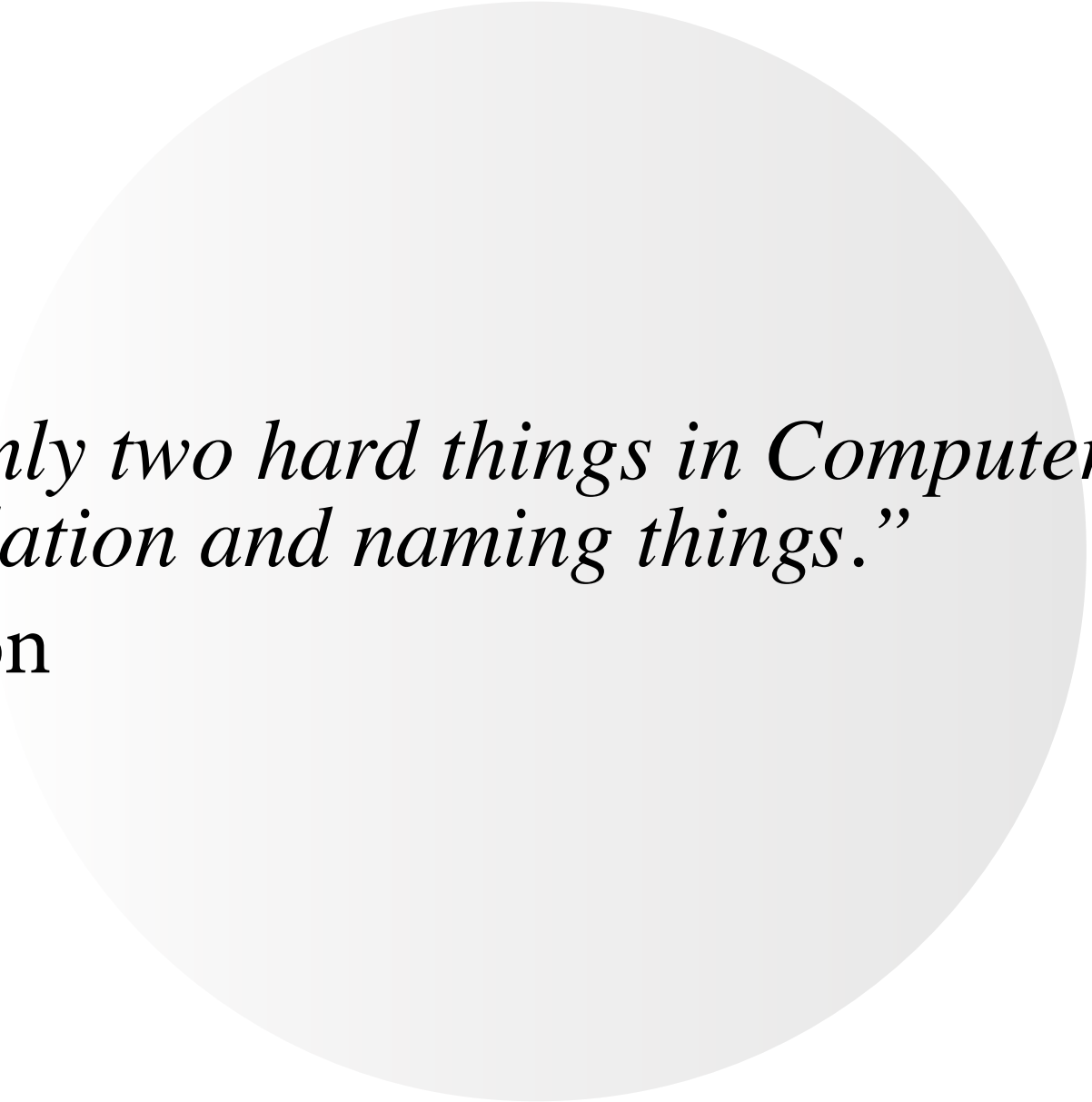


Cache-Control: A Survey in Fast Delivery Across CPU and Web





*“There are only two hard things in Computer Science:
cache invalidation and naming things.”*
– Phil Karlton

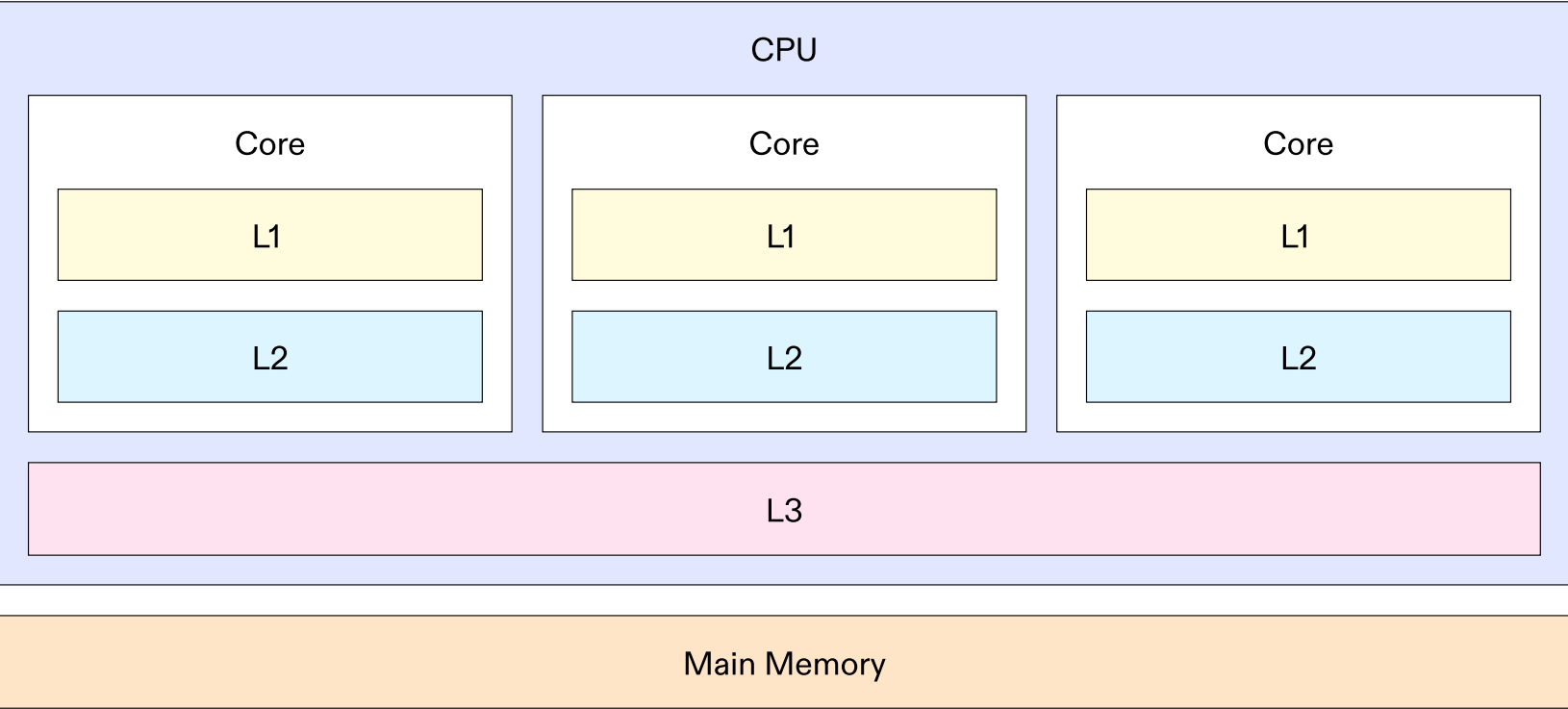
Caching is important.

In the digital age, the paradigm of caching—preparing high-latency data ahead of time, ensuring it can be retrieved instead locally at low latency—is a problem of paramount importance. In the CPU, where uncached memory reads cost precious clock cycles, the answer results in a complex architecture of multi-level caches and specialized hardware for predicting memory accesses ahead of time. On the web, caching has led to a huge economy of Content Delivery Networks (CDNs), where webpages are copied and hosted across the globe, as close as possible to the end-user, to prevent lengthy network calls back to the origin server.

CPU

Hardware caches inside of the CPU prevent main memory reads by putting the data that might be accessed soon in a place local to the CPU itself, available with much lower latency.

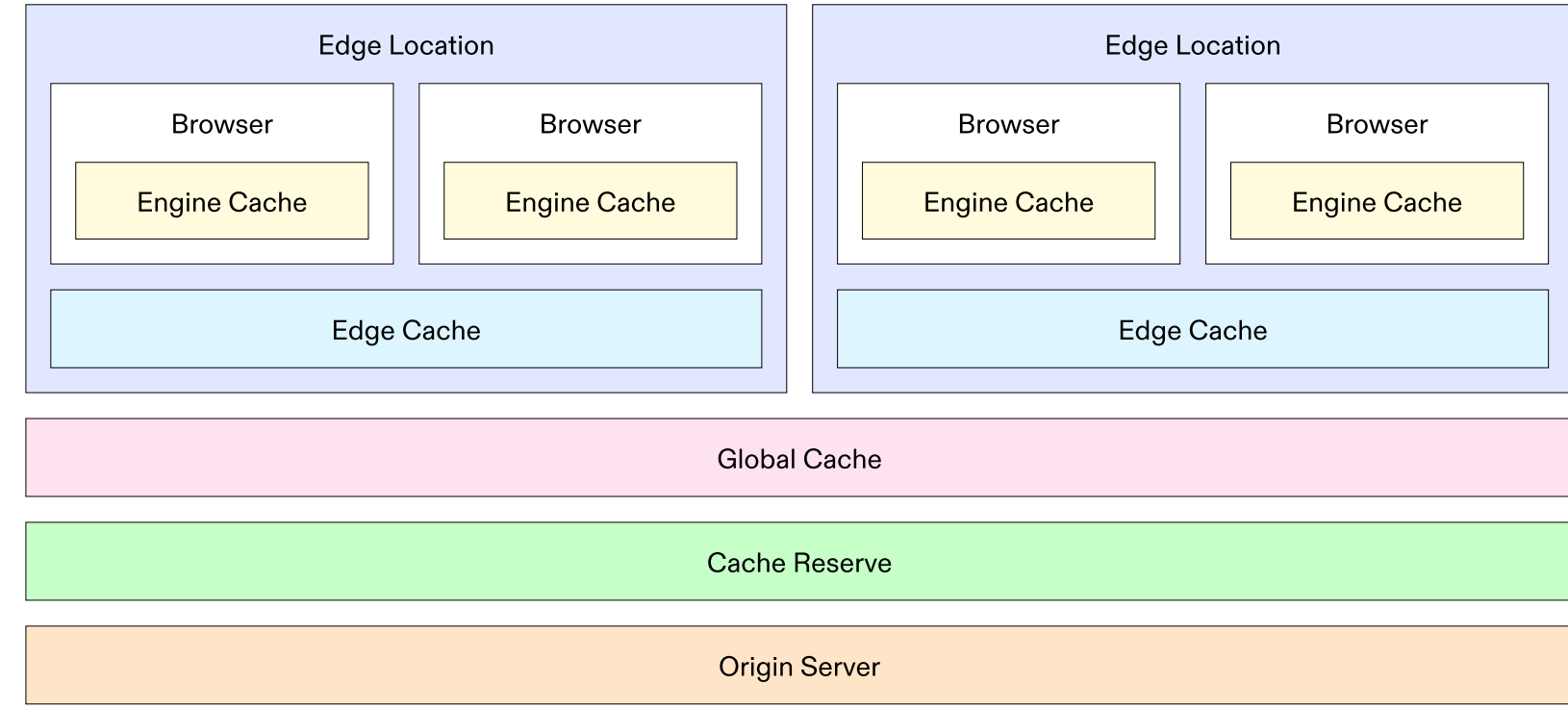
Example



Web

Instead of preventing memory reads, web caches serve to avoid network connections, egress fees on static files, and server-side content building. Caches for web resources are intrinsicially larger in size, often globally distributed, and higher latency.

Example



Okay, so...

There is a bit of area for cross-pollination in caching across CPUs and web caches—largely in the forward direction. Specifically, I am interested in the concept of branch prediction.

Branch Prediction...

To further cut back on load speeds, one can analyze data flowing through a cache and use that data itself to keep the cache hot—ready with data that could follow from the existing location.

...in the CPU.

Memory addresses of branching instructions are fed to a branch predictor, which makes a “best guess” of instructions to load after the branch, before a direction is taken.

...for the Web.

URLs in HTML can be analyzed while streaming to the user to prefetch cache resources needed on the existing page, as well as potential future pages a user might visit.

Prior Art



Prefetch URLs

CloudFlare offers prefetching by way of Link HTTP response headers. This means links for prefetchable content must be manually provided by the origin server, rather than being content-aware. This is complicated to implement.

<https://developers.cloudflare.com/speed/optimization/content/prefetch-urls/>

In my initial exploration, the only similar features I could find across all of the major CDN providers were the following:



Predictive Prefetching

Akamai is the only CDN provider I could find which offers a CPU-similar method for predictive prefetching. Their implementation operates based on historical data of requests made by the user, rather than HTML content itself.

<https://techdocs.akamai.com/property-mgr/docs/prefetching>

Deliverables

Deliverable 1

Paper

A survey paper comparing caching techniques between the CPU and the web. This will touch on topics including cache levels, eviction policies, and hardware.

Deliverable 2

Website

A web-based tool for natural language explanations of your own web caching policies (made available in HTTP Headers like `Cache-Control` and `Expires`).

Deliverable 3 (Reach)

CDN

A custom CDN layer implementation that builds upon CPU caching techniques (branch prediction) to pre-cache links found in streamed HTML responses.



Thank you!