COS 316 Precept #7:

Caching

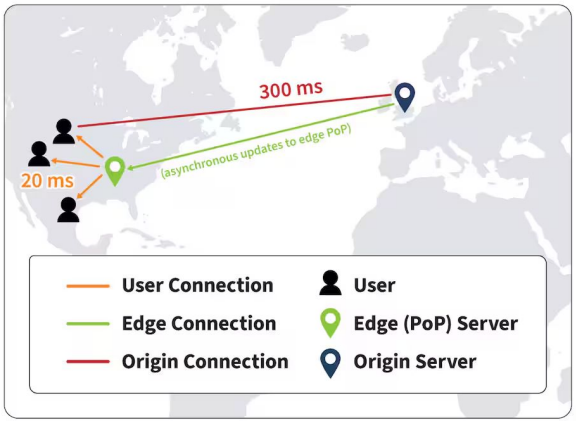
+

Eviction

- In today's precept, we'll go over some cache-related concepts that were touched on in lecture and that students need a bit of help with
- We've talked a bit about CDNs and "the edge", so we'll give a quick definition of those, before jumping back to more caching stuff

What is the Edge?

- Move compute and storage closer to end-users
- An evolution of traditional datacenter-centered architecture

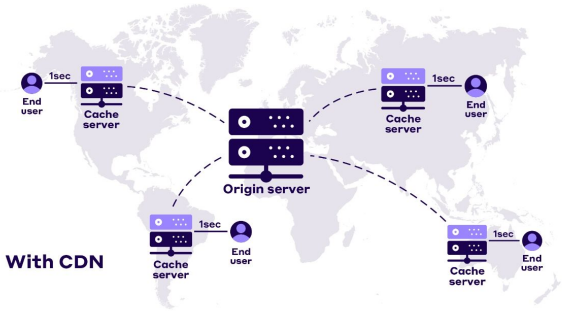


- Let's talk about the edge
- There is nothing special about an edge network
- There's no special way to communicate with them and they don't operate on some brand new layer in the network that we haven't discussed yet
- Edge networks are just a different way of organizing servers
- Rather than a provider having a single datacenter that contains a bunch of servers that handle all requests, that provider can geo-distribute those servers so that they're closer to end-users
- When we talk about the edge, we're just talking about servers that don't reside in a main datacenter and that have been placed closer to end-users

What is a CDN?

Content delivery network (CDN):

- A group of servers over a region/the world to speed up content delivery.
- CDN servers cache web content such as image and videos.
- It sends to the users who request the webpage.



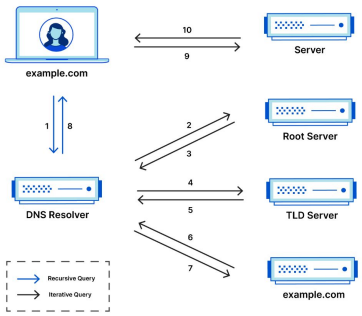
https://nitropack.io/blog/post/cdn-beginners-guide

- CDNs are groups of servers that are meant to serve content to end-users
- They’re entirely architectured around the need to serve content to geo-distributed end-users
- This means that there will be cache servers scattered around the world so they can serve content to users that are close to those cache servers

DNS

Domain Name System (DNS):

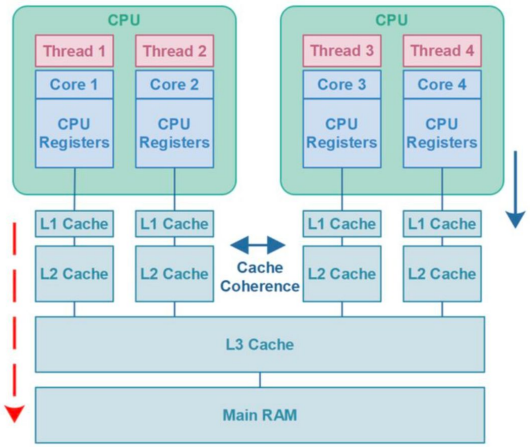
1. Send the DNS query to the DNS resolver
2. Resolver queries the root server
3. The root server responds with address of TLD server (“Here is where you should go for URLs that end in .com”)
4. Resolver queries TLD server that handles “.com” URLs
5. TLD server responds with address of authoritative name server (“Here is the server that should hold the IP address for example.com”)
6. Resolver queries authoritative name server
7. Authoritative name server responds with the IP address of example.com
8. Resolver returns response to the client



https://www.cloudflare.com/learning/dns/what-is-a-dns-server/

- Remember that DNS is the system that translates URLs to IP addresses
- DNS is a hierarchical system where fully resolving a query may take many round trips
- Let’s walk through an example of a DNS query and its ensuing response
- First, let’s talk about the DNS resolver
- In the slides, Wyatt labels this the local DNS server
- This is the machine that a client issues a query to and that ultimately returns a response
- It does all of the work
- Now, let’s move onto the actual process of issuing a DNS request and getting a response
- Look at all of these messages that must be exchanged for a single DNS query to be satisfied
- Each of those messages take time and it would be useful to cache their responses!

Example: CPU caching



Hierarchical structure

- L1 Cache: Smallest and fastest
- L2 Cache: Larger but slower than L1
- L3 Cache: Larger but slower than L2
- L1 cache 100x faster than DRAM

https://www.linkedin.com/posts/gopal-chakraborty-b6a7b91b_cpu-systemdesign-performance-activity-7191333397699436544-uxFs/

- Here is an example of a cache that you all use every day, but whose inner-workings you may not be familiar with
- This is the memory hierarchy for a CPU with a 3-level cache
- You can see that there are 3 levels in the cache, L1, L2, and L3, and Main RAM at the bottom
- Each of the caches should hold a subset of the data in the next cache
- So L1 contains part of the data in L2, and so on
- The cache closest to the CPU, the L1 cache is both the smallest cache, due to space constraints, and the fastest cache, because it's so close
- The next cache, L2 is larger because its further away from the CPU and has more space to be placed, and slower, because its further away from the CPU
- This shows that even on your local machine where data doesn't have to go across a network, caching is useful

Example: Communication Channel

TLS session caching

- Reduce connection time
- Cache key information for TLS connections
- Length of time for the information being kept would be set

TCP connection caching (TCP fast-open)

- Speed up the opening of successive TCP connections between two endpoints
- Generate a TFO cookie when establishing connection for the first time.
- When re-connecting, the client can append the cookie to the SYN packet to authenticate itself. The server will directly send data without waiting for finishing the 3-way handshake.

Belady's Algorithm

- Provably optimal
  - Will maximize hit rate and minimize miss rate for any given cache
- Requires knowledge of the future
- Impossible in practice!

- We've talked about a few cache eviction policies
- Remember that these come into play when a cache is full, something needs to be inserted into the cache, and the cache needs to decide what to evict
- We've discussed the FIFO algorithm, which will evict the oldest object in the cache
- We've discussed the LRU algorithm, which will evict the item that was last used the furthest in the past
- We've discussed the LFU algorithm, which will evict the item that has been used the least frequently
  - We'd need to keep an access count for this policy
- Each of these may win out over the others, depending on the data access pattern of the clients that are using the cache
- But is there some optimal algorithm? Some algorithm that is the best that you can ever do with a cache? Yes!
- That is Belady's algorithm
- If a cache were to employ Belady's algorithm for eviction, it would be get the highest hit rate and lowest miss rate that's possible for that cache
- So why isn't it employed? Because it requires knowledge of the future!
- For Belady's algorithm, the item that the cache decides to evict is the item whose next access is the farthest in the future