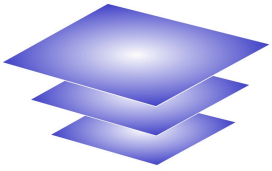
Agenda

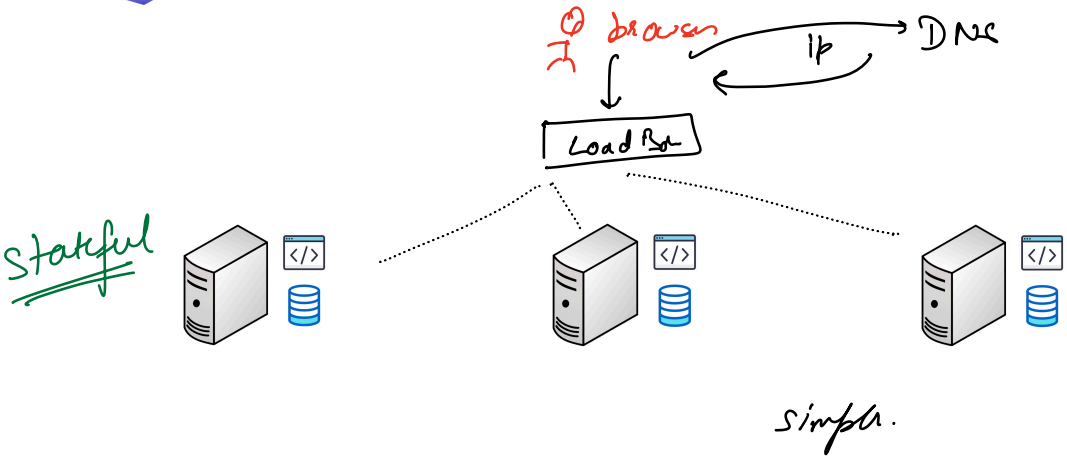
1. Appserver Layer
2. Caching
3. Invalidation & Eviction
4. Local Caching

support@scaler.com

10:30



Appserver Layer



Is this good design?

1. Data & business logic are tightly coupled.
2. for any code deployment, data temporarily unavailable.
3. webserver
   - CPU
   - RAM
   - fast n/w

   database
   - disk space
   - fast I/O

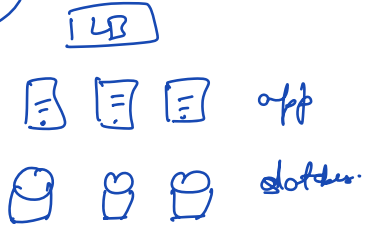
   Server is expensive

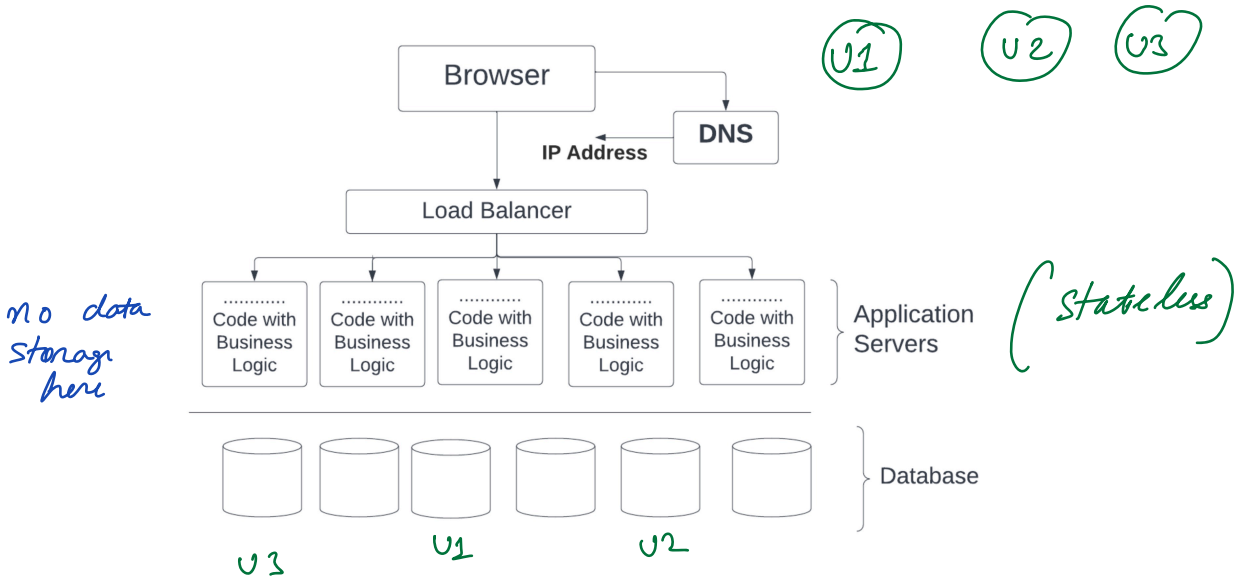
   webserver & db are competing for resources.

Split it!

Cons

1. complex (buy more albeit specialized hardware)
2. additional latency.





? Which request to which server? No. ∵ app servers are stateless

exceptions → authentication & authorization (session) GoTrue

(cases when you need stateful)

→ maintain local cache

Who does the Consistent Hashing? Load Balancer / App Server

✗ stateless app servers

✓ ∵ database is stateful & sharded.

hbase client
MySQL client
→ implement consistent hashing internally.

How do the Appservers know how many & which DB servers are available?

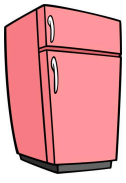
Zookeeper

cluster manager

Making Tea

? Ingredients — tea leaves / powder, water, milk, sugar, Earl grey tea

supermarket



fridge — caching layer.

→ slow access

→ source of ingredients V.V. wide selection

→ always has fresh items available

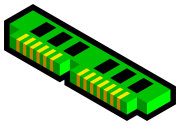
→ access is very fast

→ limited in space only have freq. used items.

→ items can go stale

multiple levels of cache

CPU / registers / L1 L2 L3 cache / RAM / HDD



Caching

1. user data
2. images / videos / static media
3. CSS / JS / Html
4. IP address

→ DNS cache
→ static media
- CSS
- JS
- HTML
- Images

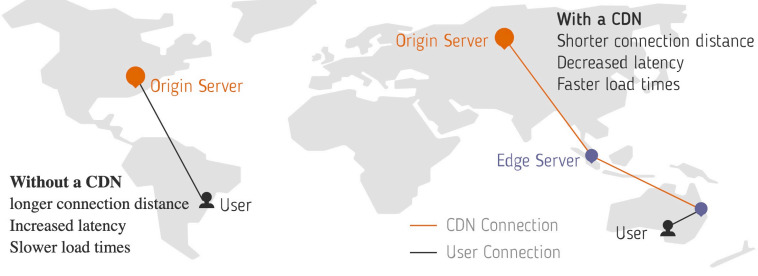
→ Cookies / Local storage / indexedDB
- username
- theme (light / dark)
- config
- profile pic
- language

India

google → USA

- Cloudflare 375,000 servers
- Akamai
- Amazon Cloudfront
- Fastly

Content delivery networks



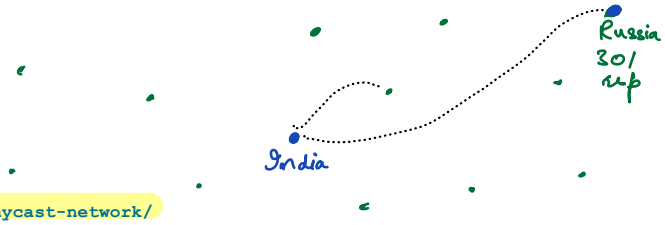
https://www.cloudflare.com/learning/cdn/what-is-a-cdn/

DNS mapping

cdn.cloudflare.com

? How to get the IP of the nearest CDN server?

↳ geo DNS (this adopted)
↳ any cast

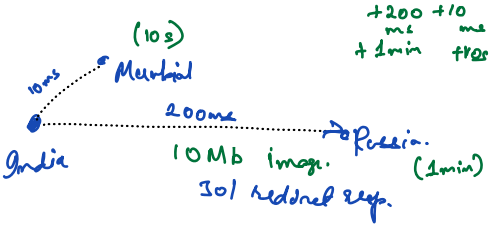


https://www.cloudflare.com/en-gb/learning/cdn/glossary/anycast-network/
https://constellix.com/news/anycast-vs-geodns

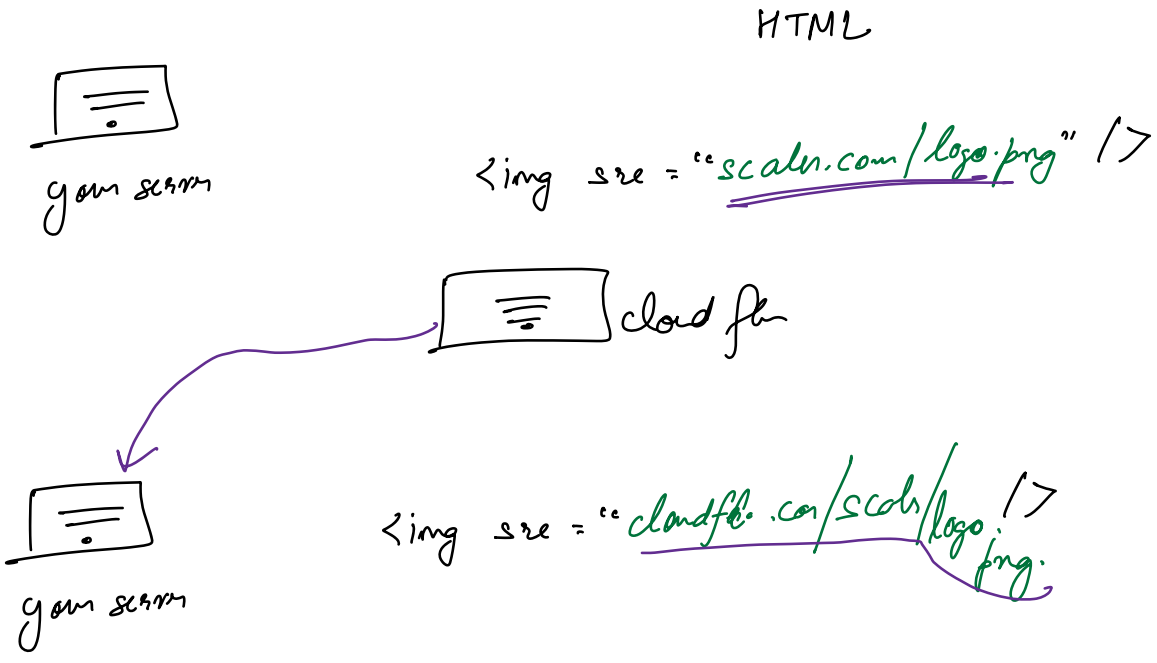
→ url → ip

Local Cache — each appserver maintains its own cache

Global Cache — Cache layer shared by all appservers.



How do CDNs get data?



Challenges with Caching

→ cache is not the source of truth (data)

→ data in cache could be stale

→ periodically invalidate stale data

→ limited in size

→ evict data from cache

Cache Invalidation

Write Through

Write Back

Write Around

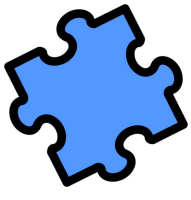
Cache Eviction

First in First Out

Least Recently Used → HashMap + Heap
↳ HashMap + DLL

Last in First Out

Least Frequently Used



For next class

? Local caching of test data on the app server. How to invalidate on the update?

? How is facebook's newsfeed calculated, and how can you make it very fast?

some hosting where media files or stored

HTML css JS

images videos

scaln.com/index.html → vercel

scaln.com/logo.png → vercel

cloudflare

resources

scaln.com/logo.png

scdn.com/

cdn.cloudflare.com/1/abe57

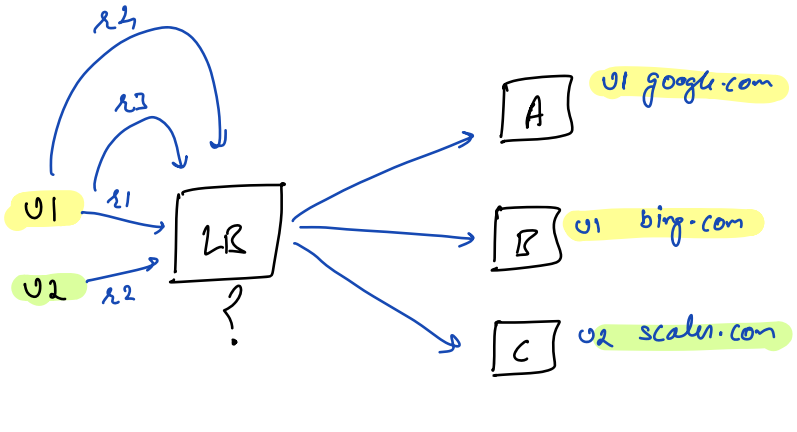
scdn.com/__

SSL

(1) multiple app-servers
(assume data is stored in app-servers themselves)

(2) data is sharded (partitioned across multiple servers)

(3) LB gets requests from users & needs to send them to an appserver.



request on bing routed randomly.
✗ bad idea

if user wants to read data LB has no idea!!

(user-id % N)
range based

the user's req go to same server
LB knows which server to send to.

servers increase or decrease.
↳ lot of data shuffling.

hash functions

$$H_1(x) = (10x^2 - 3x + 1)\,\%\,100$$

$$H_2(x) = (x^3 + 3x^2 - 2x)\,\%\,100$$

$$H_3(x) = (3x^2)\,\%\,100$$

	H_1	H_2	H_3
S_1	37	2	10
S_2	45	99	73
S_3	0	11	58

	H_1
U_1	77
U_2	38
U_3	12
U_4	56
U_5	22

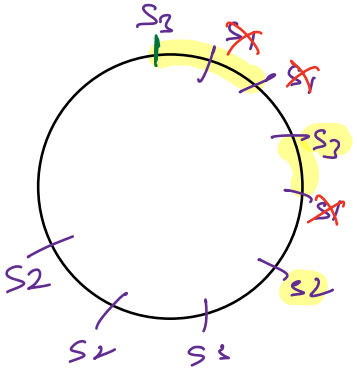
Load Balancer

① Health check / heartbeat to find out what all servers are alive.

S_1 S_2 S_3

②

0	2	10	11	37	45	58	73	99
S_3	S_1	S_1	S_3	S_1	S_2	S_3	S_2	S_2



③ U1 req.

$H_1(U1) = 77 \longrightarrow S2$

U2 req.

$H_1(v_2) \rightarrow S2$

v_3 req

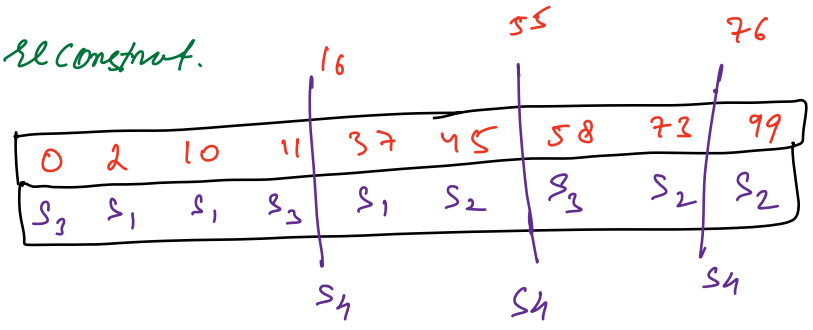
$H_1(v_3) = 12 \rightarrow S1$

(4) add new server S_4

Load Balancer detects that S_4 is also free.

	H_1	H_2	H_3
S_4	76	16	55

Reconstruct.



H_1

v_1 77 → S2

v_2 38 → S2

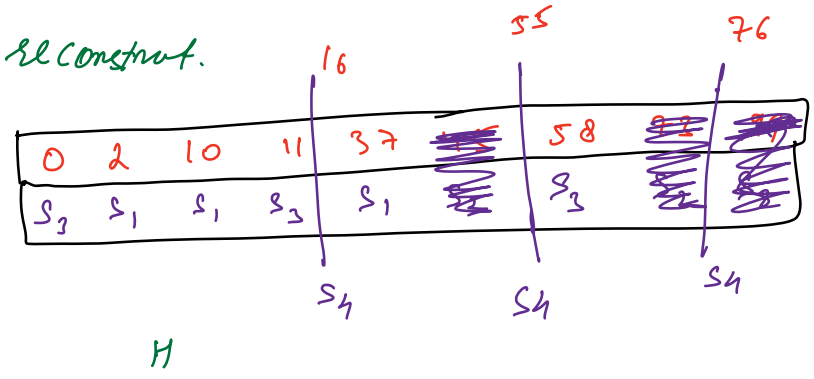
v_3 12 → earlier S1 now S4

v_4 56 → S2

v_5 22 → S1

(5) S2 goes down.

load balancer detects that S2 is down



U_1 77 → S2

U_2 38 → S2

U_3 12 → earlier S1 now S4

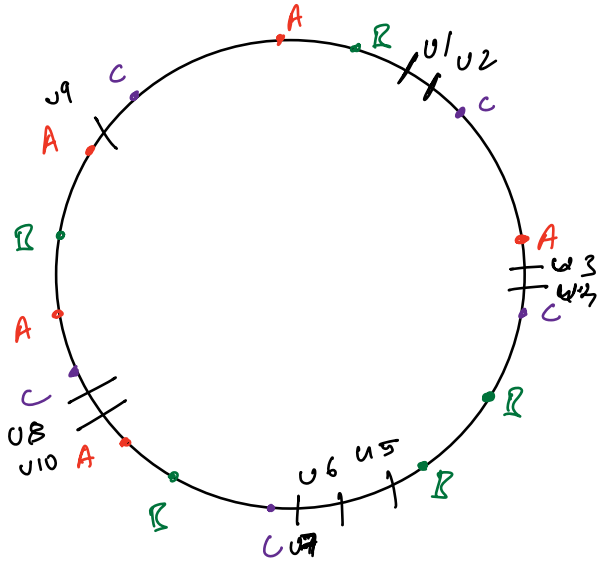
U_4 56 → S3

U_5 22 → S1

S3
S4
S4
S3
S1

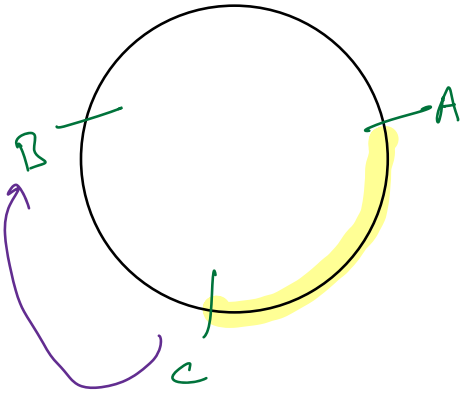
A, B, C

5 holes



10 USB → C

U1 → A
U2 → A
U3 → B
U4 → B
5 → B
6 → B
7 → B
8 → A
9 → A
10 → A



with 1 hash

load of S3 → first one ser.

100 servers

each ser 1% of total load

S43 goes down → S57

1% + 1%

1 ~~43~~ 56 57 58 — 100

cascading failure

(1 hash)

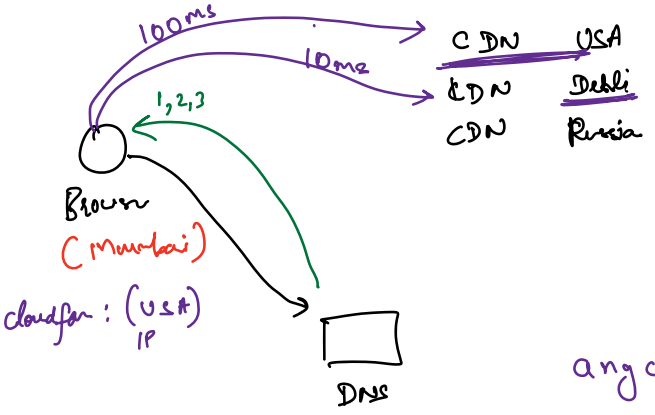
1% 1%

S43 (16 hashes) → 16 diff servers

99 - 16 ⇒ 1%

16 ⇒ 1% + 1/16

1.0625 %



anycast