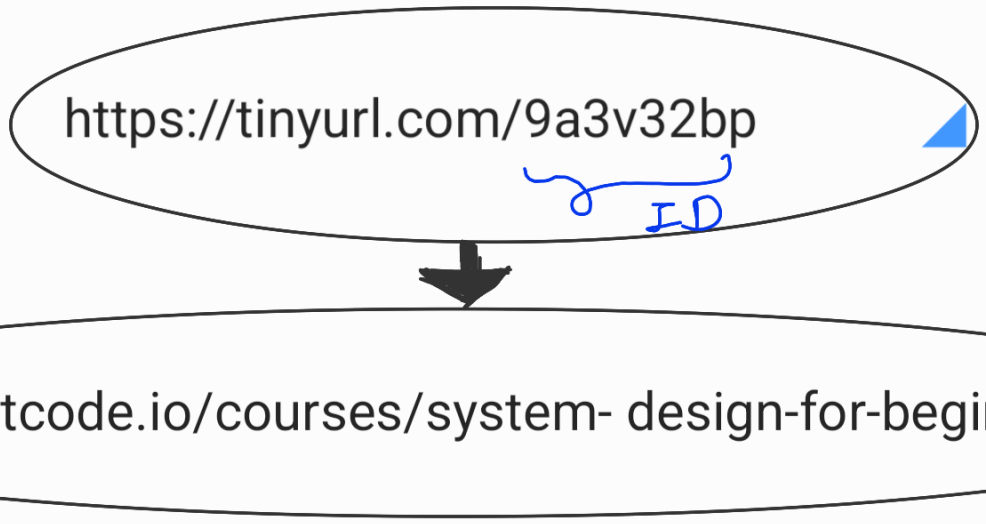
(To be opened in dark Mode)

Design

TinyURL



So, we will be keeping a track of :-

↳ Short URL ID (8 chars)

↳ owner of the URL (create & delete)

↳ where it redirects to.

↳ time period to expire

(1 year default)

Q What should happen if multiple users are shortening a same URL?

Ans

We can't give one same short URL to all since then problem of ownership will arise.

So, we need to create a unique short URL every-time.

Q// Will we do more reads or writes?

Ans Read :- Everytime one opens a URL.

Write :- Every time one creates a short URL.

∴ No of reads >> No. of writes.

* As per interviewer :-

100 reads : 1 write

And a

1 Billion short URL per month.

and 8 char short URL I.D.

So to make 1 Billion possible.

I will use

- 0 to 9 ⇒ 10
- a to z ⇒ 26
- A to Z ⇒ 26

62

62^8 total URL's

$\approx$ 1 Trillion

$\approx \frac{1 \text{ trillion}}{1 \text{ Billion / month}} =$ 1000 months

$\approx$ 80 years, we won't be short of URL's.

Moreover

1B / month

$\approx$ 33 M / 30 x 24 x 60 x 60 sec

= 400 writes / sec

⇓

40,000 reads / sec.

Storage

↳ 8 bytes for the Shortened ID.

↳ 256 bytes for the long URL

↳ 50-60 bytes Misc. meta-data (UID, date of expiry)

$\approx$ 300 bytes per write

$\approx$ 300 Billion Bytes / month

$\approx$ 300 GB / month

H.L.D

* Since, we will be having 100 x reads than writes.

We choose

↳ No SQL D.B.

↳ Because

↳ No ACID property

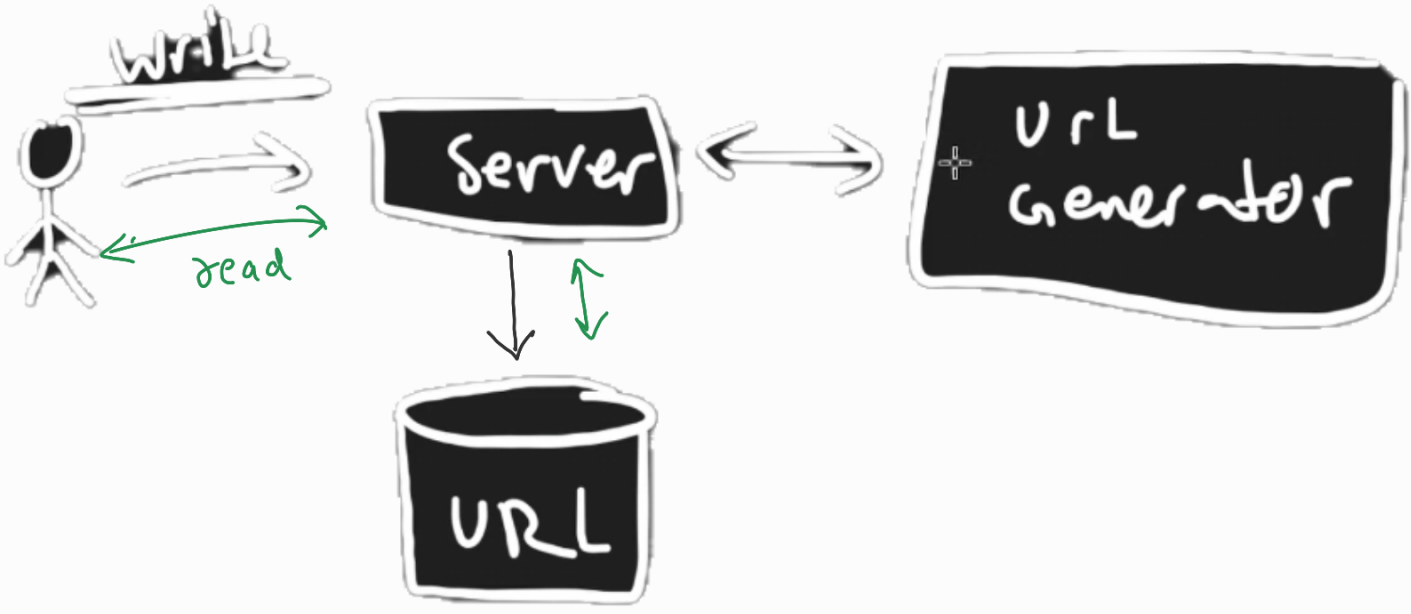
↳ Designed to handle scalability

↳ No ACID
↳ No RDBMS.

} we do not need these. as neither we are joining nor querying a lot.

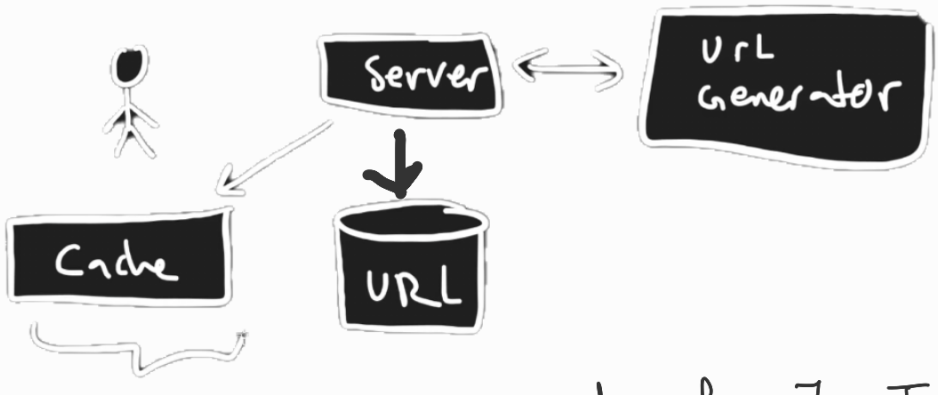
↳ And no SQL D.B gets in Sync over some time and here writes are much

lesser than reads.



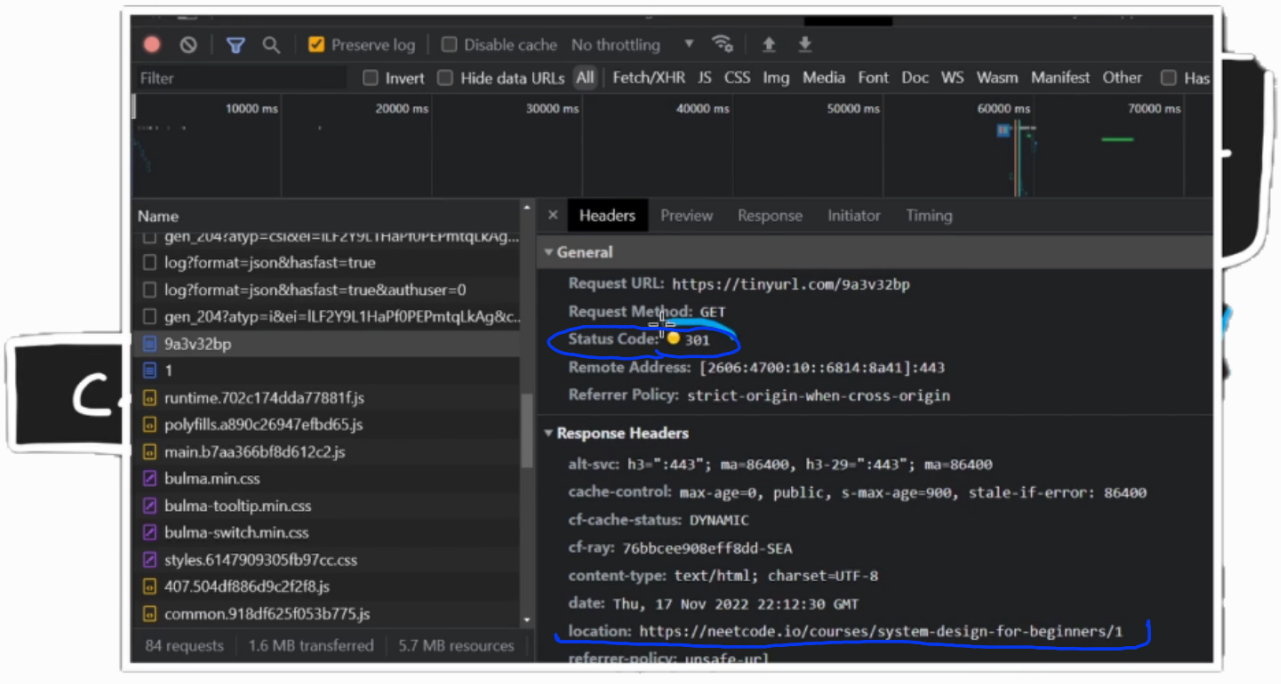
Moreover, since some URL's are going to be a lot more popular than another, so I should add a caching layer in b/w with LRU a Cache. LFU won't be used because one URL might become highly used when a trend was going on and then never used.

Design TinyUrl



in memory (out of 1 TB total URL's, we require only few into memory, so size ≈ 256 GB)

But how will the re-dirⁿ of the URL works?



when we hit, tiny URL /something it returns with '301' (resource has been moved) and a response header having location column of the same resource.

Now, it becomes the responsibility of the browser to redirect us.

301	302
* Permanently moved	* Temporarily moved.
Browser will cache the result.	* Everytime hit tiny URL
* Used to reduce load on the server.	* Can be used for analytic's purposes or if the expiry is quite near.

H.L.D for Url generator.

(i) A. D.B with all keys possible.

↳ Produce it beforehand.

(ii) Keeping 10 k keys in the memory which are not used yet. and allocat from these and when I am about to run out of keys, then get some new keys.

iii) Mark the key in the D.B as used.

D.B schema for url gen.

key	varchar(8)
used	bool.

Now, issue arises when 2 users comes at same time and asks for the key and gets the same key

So, will use SQL database as is ACID, isolation property will help here. to prevent from same key being allocated twice by starting and stopping transaction.

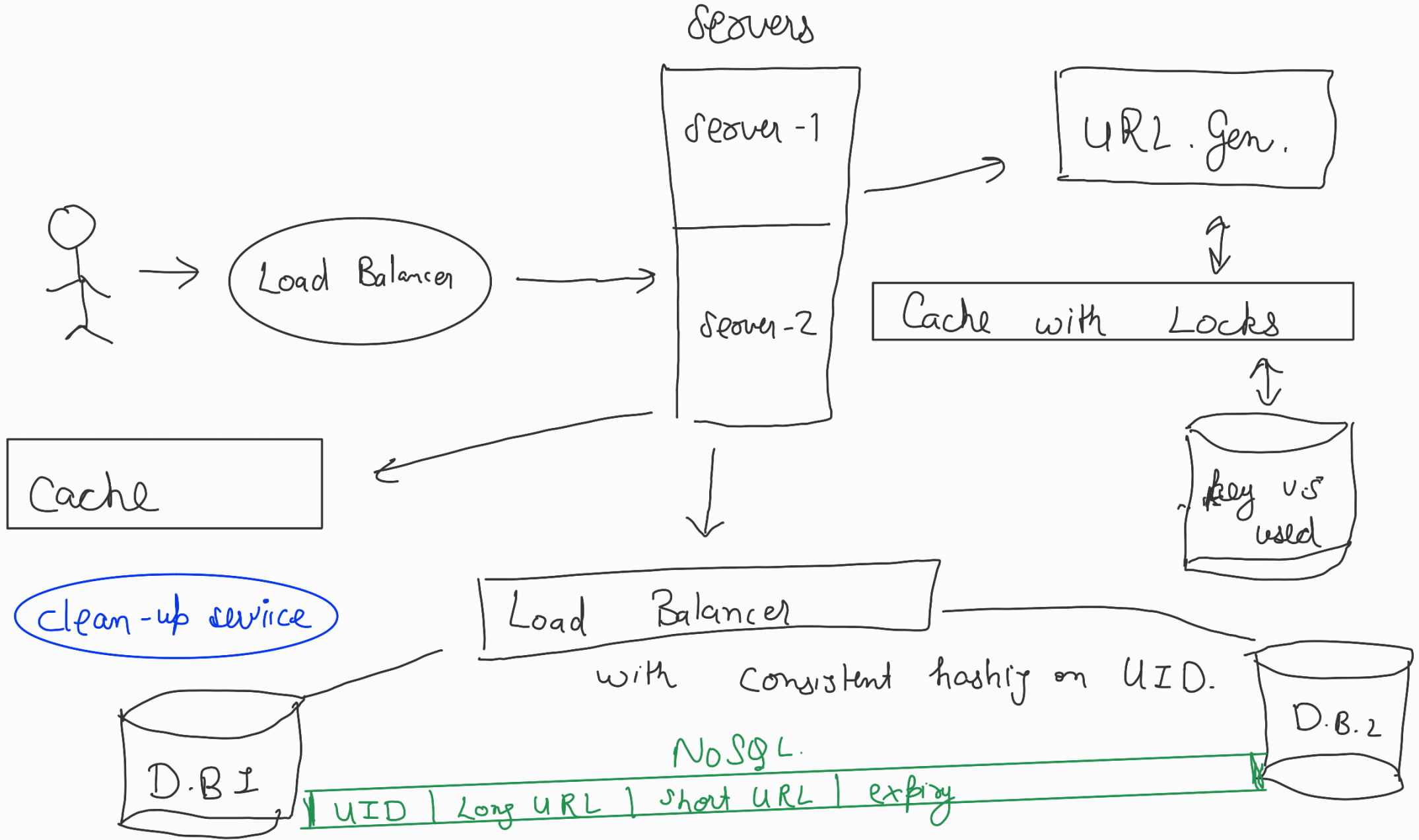
But in the above approach, we will loose up the caching benifits

So, we will implement the locking service with cache.

How URL expiry will work?

It won't be end of the world if the URL remains one 1 day extra in my system.

So, everyday, I will run a clean up service at say 00:00 to check for expired keys, delete them D.B and mark as unused in the URL Generator's D.B.


Servers
Server-1
Server-2
URL. Gen.
Load Balancer
Cache with Locks
Key vs used
Cache
clean-up service
Load Balancer
with consistent hashing on UID.
D.B.1
D.B.2
NoSQL.
UID | Long URL | Short URL | expiry