URL Shortener

Saturday, 26 September 2020 4:48 PM

APPROACH

- Gather Requirements.
- Estimate Scale
- Design the schema
- Write high level APIs
- Scale the system.

Design URL shortner

Step 1: Gather Requirement

X - Analytics required? how many times opened?
X - support customised URL?
  ↳ /scaler
✓ - persist for how long? Expiry?
- shortening capability.

Step 2: Traffic & data

- How much Data (single or multiple) sharding?
- Read heavy or write heavy
  [QPS] < Read / write
- No. of machines
- Data access pattern.

→ LinkedIN will have much higher scale.
- 10M URL shortened every day.
- Every URL is clicked 10-50 times.
- HOT URLs → caching (Modi ji)

10M X 365 X 10 = 36.5 B URL in 1 year

- length of shortend URL = x ⇒ $26^x > 36.5B$
  ∼ 7-8 chars.

(a) size of 1 entry: (7, 200) ~ 200 B

size in 1 year = 200 X 36.5 X 10^9
= 7.3 TB ← not in a single machine.

(x) Max data in a single m/c = 1 TB

(b) Every URL is read 10-50 times

(c) Estimate QPS = write = 10 M / day
= $\frac{10}{24 \times 3600}$ M / sec
∼ 1000 QPS

read = write * 50
= 5000

(d) Read Heavy system

Step 3: Design Goals

- Consistency X
- Availability ✓
- Latency very low
- Resiliancy Multi A-Z

Step 4: Single System

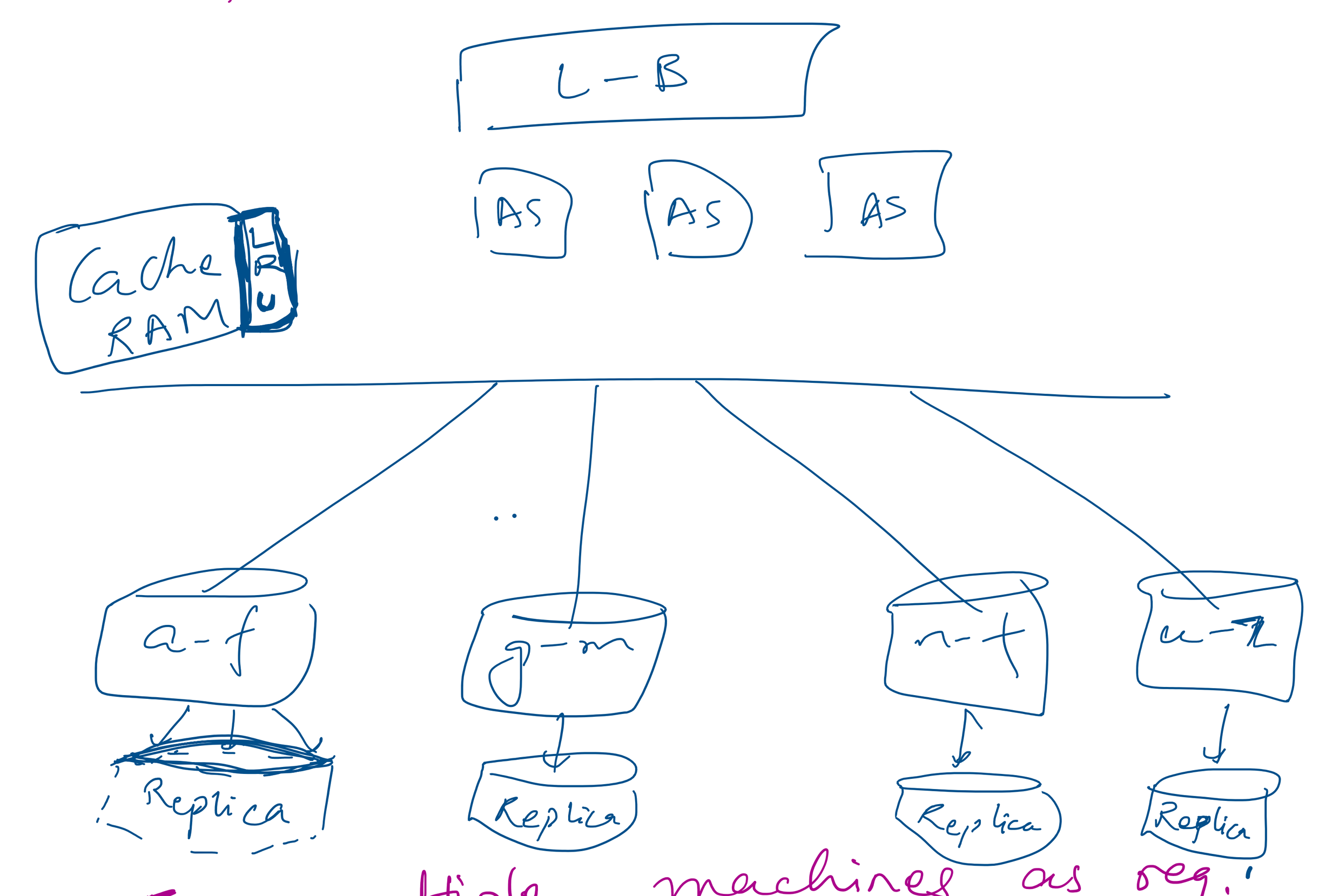
API: a) create (url, expiry)
  ↳ return shortened_url.
  Cons.
  - expiry_time
  - analytics
  Pros.
  - save_space

option 1: • Store next 10M strings in an array. Generate a random number between 0 to N. Replace element at random pos to N. and then generate from 0 to N-1.

owner_id	orig-url	short_url	expiry_url.

b) Delete (short-url): Auth-header
c) // Analytics APIs
d) Read (short-url)

→ read from cache
- if not tnt then fetch from DB.



Step 5: multiple machines as req. by scale determined by step2.

SEARCH TYPEAHEAD
AUTOCOMPLETE

step 1
- alphanumric characters
- suggest term/phrases
- personalisation X
- 5 suggestion
- suggestions based on frequency.
- spell checking X
- suggest only after 3 chars.

step 2
- 1 Billion search queries
- autocomplete queries: 5 X 1 B
  = 5 B reads
- writes? : 1 B writes
- QPS = $\frac{6 B}{86400}$ ≈ 70K
  (100 - 200 machines)
- Data size: ∼ 130 TB

query Avg: 750 chars
10 % → new (create)
90 % → old (update)

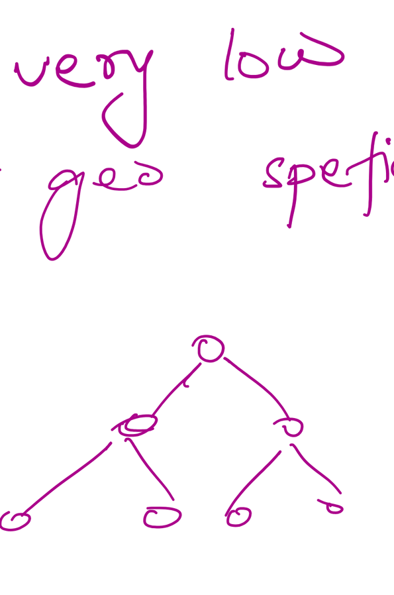
Query	Count

150 X 100M = 15 B
= 15 GB / day

= (15 X 365 X 25) GB
= 135 TB

- Read > write

step 3:
- available
- very low latency
- geo spefic.



26
26 X 26
26 X 26 X 26
26^4