Design TinyURL System

Practice on: 17 Jan 2025 - Friday

Time Taken: 80 Mins (1 Hour 20 Mins)

Qs for the interviewer

1. What can be length of the tinyURL?
2. System is for a B2B or B2C?
3. DO we need to rate limit a user - how many tinyURL an user can create in a min?
4. Can user provide custom alias in the tinyURL?
5. Can user provide custom expiration for requested tinyURL?
6. How long system/database need to persist the tinyURL mapping?
7. Total users? DAU?
8. What will tinyURL consist of? sample? Can we have special characters?

Requirements

Fuctional Requirements

1. User provides a long URL and get a short URL.
2. User click on the shortURL and get redirected to the longURL
3. User can provide custom alias and expiration for the stored URL (OPTIONAL)

Non Functional Requirements

1. System should have low latency (return short URL under 200 ms)
2. System, should be highly available
3. System should be durable - stored URL should not be lost

Capacity Estimation

Assumptions:
Total Users: 100 million
DAU: 10 million
Read:Write ratio: 10:1
LongURL avg length: 100 characters
ShortURL avg length: 10 characters

Throughput - RPS & WPS
WPS = 10 million * 1/10 = 10^6 => 10^6/24*60*60 ~ 12WPS
RPS = 12WPS * 10 = 120 RPS

Storage:
stortURL = 10 bytes longURL = 100 bytes created_date = 10 bytes
Aprox ~ 150 bytes for every entry
Storage for a day: 10 million * 1/10 = 1 million writes * 150 bytes = 15 * 10^9 => 15 GB
Storage for 10 years = 15 GB * 365 * 10

Core System Entity

1. User
2. LongURL
3.ShortURL
4. Alias
5. Expiration

API

1. **User submits a longURL -> gets a shortURL in response**

REQUEST:

HTTP POST API /short
body{
long_url: "www.facebook.com/profile/angelPriya",
alias: "",
expiration: ""
}

RESPONSE:

{
short_url: "www.tiny.com/qb2323j"
}

2. User clicked on tinyURL -> gets redirect to longURL

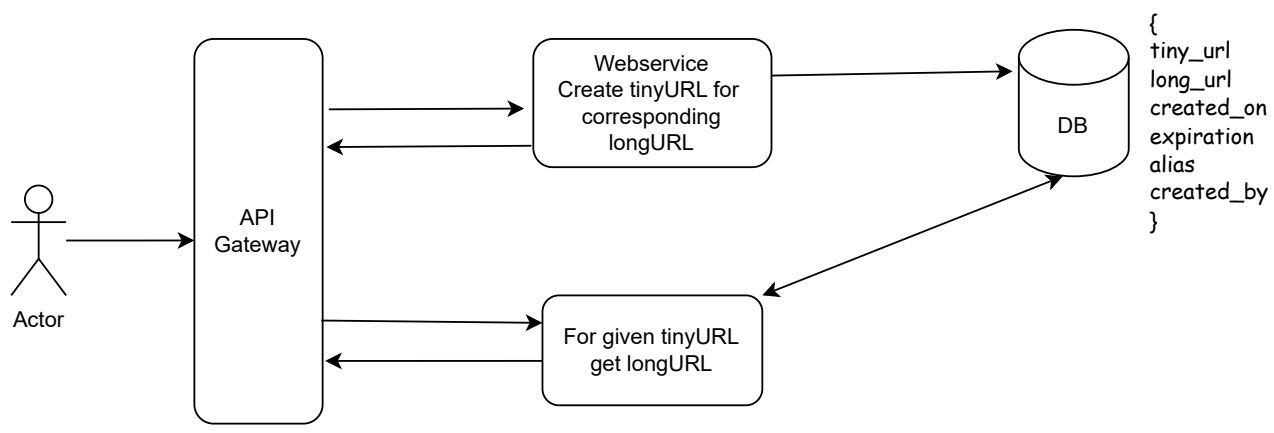
REQUEST:

HTTP GET API /qb2323j

RESPONSE:

Status code 302
{
redirect_url: "www.facebook.com/profile/angelPriya"
}

High-level-system-designe



Deep-dive

[1] Creating tinyURL for corresponding longURL

1. One way to create tiny URL can be using SHA-256 on longURL to get a hash. This hash will be long so can use the first 7 characters as the key of tinyURL
Drawback :
> their can be collisions: as first 7 char hash can be same - counter can be used but we before saving in DP we need to check the uniqness of the key each and every time.

2. We pre-calculate a lot of hash key and keep them stored in another database, whenever a new longURL request

comes we get one unused key and map that with the long user and store in the mapping DB
Pro:
> NO need to calculate/find key on the fly > this will reduce the latency
> No need to check the uniquness of the key

In unique_tinyUrl_key table we can have a boolean value that can be set false whenever its been used
Or we can have 2 DB > one will keep unused keys and other will store the used key > this is help in increasing the latency.

[2] How long can the tinyURL hash Key

12WPS > 12 * 60 * 60 * 24 ~ 1 Million tiny URL created every day
[a-z] + [A-z] + [0-9] > 26 + 26 + 10 > 62 characters
6^62 ~ ????
7^62 ~ ????

This way we can decide we the tinyURL's min length

[3] Low Latency

> Introduce cache before DB
> Once a tinyURL created probability of its been used in a short time is high, so we keep in cache
> Once a tinyURL been read from DB, put that in cache
> LRU will be the eviction policy
> caching strategy ?????
> DB stores to cache?? OR web service stores to cache??

[4] Scaling

> We can do horizontal scaling > more number of web sevices for creating tinyURL
> Multiple replication of DP can be kept near to the user
> DB partition can be done on the basic of alphabetics of the tinyURL for DB scaling

[5] Database

> better to have noSQL DB as data need to be replicated across diff Data centers
> Master slave architecture can be used

Final Design