TinyURL Design

Functional requirements

- Create shorter and unique aliases for long URLs
- Users can be able to customize the short URL
- Users can provide an Expiration time

Nonfunctional requirements

- Security – Limit the number of requests per user thus preventing users from denial of service attacks, either through User-Id for registered user or IP address for anonymous users.
- Availability – TinyURLs should be available even when system is down
- Redirected with minimum latency

Estimates

- Traffic
  - We assume
    - Read : Write ration of this system is 200:1
    - 500 million new requests/month
  - Therefore, 500 * 200 million READ requests/month

500,000,000 * 200 READ	requests/month
(500,000,000 * 200) / (30*24*60*60)	requests/second
100,000,000,000 / 2,592,000	requests/second
38 k	requests/second
~ 40k READ QPS	
~ 200 CREATE QPS	

- Storage
  - We assume,
    - each CREATE URL request takes 500 bytes
    - each user information takes 200 bytes
    - number of users 200 million/year

  - User info storage
    200,000,000 users/year * 200 bytes/user
    40,000,000,000 bytes/year
    40 TB/year

  - Request storage
    500,000,000 requests/month * 500 bytes/request
    250,000,000,000 bytes/month
    250,000,000,000 *12 bytes/year
    3,000,000,000,000 bytes/year
    3 PB/year

- Bandwidth
  - We assume
    - One request per user
  - We know,
    - each CREATE URL request takes 500 bytes
    - each READ URL request takes 200 bytes (Does READ actually take memory?)
    - 40k READ QPS
    - 200 CREATE QPS

  - CREATE URL requests
    500 bytes/request * 200 requests/second
    100,000 bytes/second
    100k bytes/second
    100 kbps

  - READ URL requests
    200 bytes/request * 40k requests/second
    8,000,000 bytes/second
    8 Mbps

- Memory
  - We assume,
    - 30% READ URL requests per day are cached

30% * 40	requests/second
30% * 40,000 * 60 * 60 * 24	requests/day
1,036,800,000	requests/day

~ 1 billion READ URL requests/day are cached

1 READ URL request takes 200 bytes
1,000,000,000 requests/day * 200 bytes/request
200,000,000,000 bytes/day
200 GB/day cached for READ URL request

System APIs

- createURL(userId, longURL, <customURL>, <customExpirationTime>)
- readURL(shortURL)
- deleteURL(shortURL)

Data Model

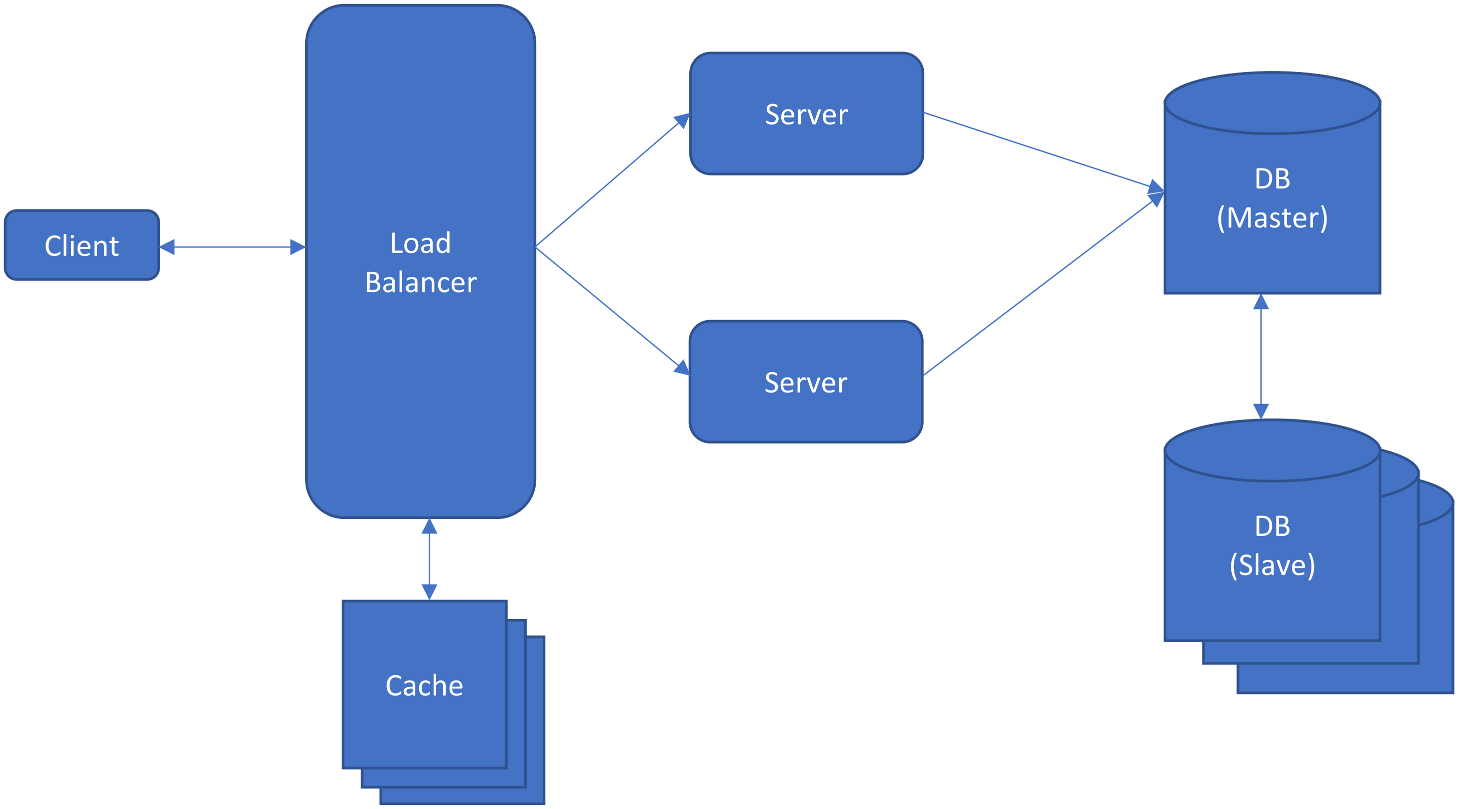
- Availability is of higher precedence than consistency

USER
UserId
Name
Email
Last Login
Account Creation Date

URL
URL Key
Original URL
Expiration time
UserId

KEYS
URL Key
isUsed

Block diagram



- I use MongoDB here as it is an AP database
- MongoDB has a Master-Slave configuration with replication
- I will provide Sharding as when some of the DBs are down, the whole system doesn't crash/ will not work.
- I will have cache at the load balancer. This caches the HOT URLs. Cache eviction policy – 24 hours