wuyichen24 / system-design-interview Public

Code Issues 4 Pull requests Actions Security Insights

master

system-design-interview / problems / Build_URL_Shortening_System.md

wuyichen24 Update Build_URL_Shortening_System.md History

1 contributor

130 lines (116 sloc) | 6.22 KB

Build URL Shortening System

Real-life examples

- TinyUrl

Requirements clarification

- **Functional requirements**
  - URL Shortening (Write): Given an original URL, our service should generate a shorter and unique URL of it.
  - URL Redirection (Read): When users access a short URL, our service should redirect them to the original URL.
  - Optional function requirements
    - URL Customization: Users should optionally be able to pick a custom short URL for their original URL.
    - URL Expiration: Shorter URL will expire after a standard default timespan. Users should be able to specify the expiration time.
- **Non-functional requirements**

    - The system should be highly available (If our service is down, all the URL redirections will start failing).
    - URL redirection should happen in real-time with minimal latency.
    - Shortened links should not be guessable (not predictable).

Estimation

- **Traffic estimation**
    - Our system will be read-heavy (Lots of redirection requests compared to new URL shortenings).
    - Read-write ratio is 100 : 1 (*Assumed*)
    - Number of read actions and write actions per month
        - Number of writes (URL Shortening) per month = 500 millions (*Assumed*)
        - Number of reads (URL Redirection) per month= 500 millions x 100 = 50 billion
    - Frequency of read actions and write actions per second (QPS)
        - Frequency of writes per second = 500 millions / (30 days x 24 hours x 3600 seconds) = 200 times/s
        - Frequency of reads per second = 200 times/s x 100 = 20000 times/s
- **Storage estimation**
    - Types
        - Data: Yes
        - File: No
    - Capacity
        - Time length of storing a record = 5 years (*Assumed*)
        - Number of records created in 5 years = Number of writes per month x Number of months = 500 million x 5 years x 12 months = 30 billion
        - Size of one record = 500 bytes (*Assumed*)
        - Total capacity needed in 5 years = 30 billion * 500 bytes = 15 TB
- **Bandwidth estimation**
    - Write bandwidth = Frequency of writes per second x Size of one record = 200 times/s x 500 bytes = 100 KB/s
    - Read bandwidth = Frequency of reads per second x Size of one record = 20000 times/s x 500 bytes = 10 MB/s

System interface definition

- **Interface 1**
  - `createURL(original_url)`
    - Function
      - Create a new shorter URL.
    - Parameters
      - original_url (string): Original URL to be shortened.
    - Return
      - The short URL.
- **Interface 2**
  - `getURL(api_key, short_url)`
    - Function
      - Get the original long URL of a short URL.
    - Parameters
      - short_url (string): The short URL to be redirected.
    - Return
      - The original long URL.

Data model definition

- **Schema**

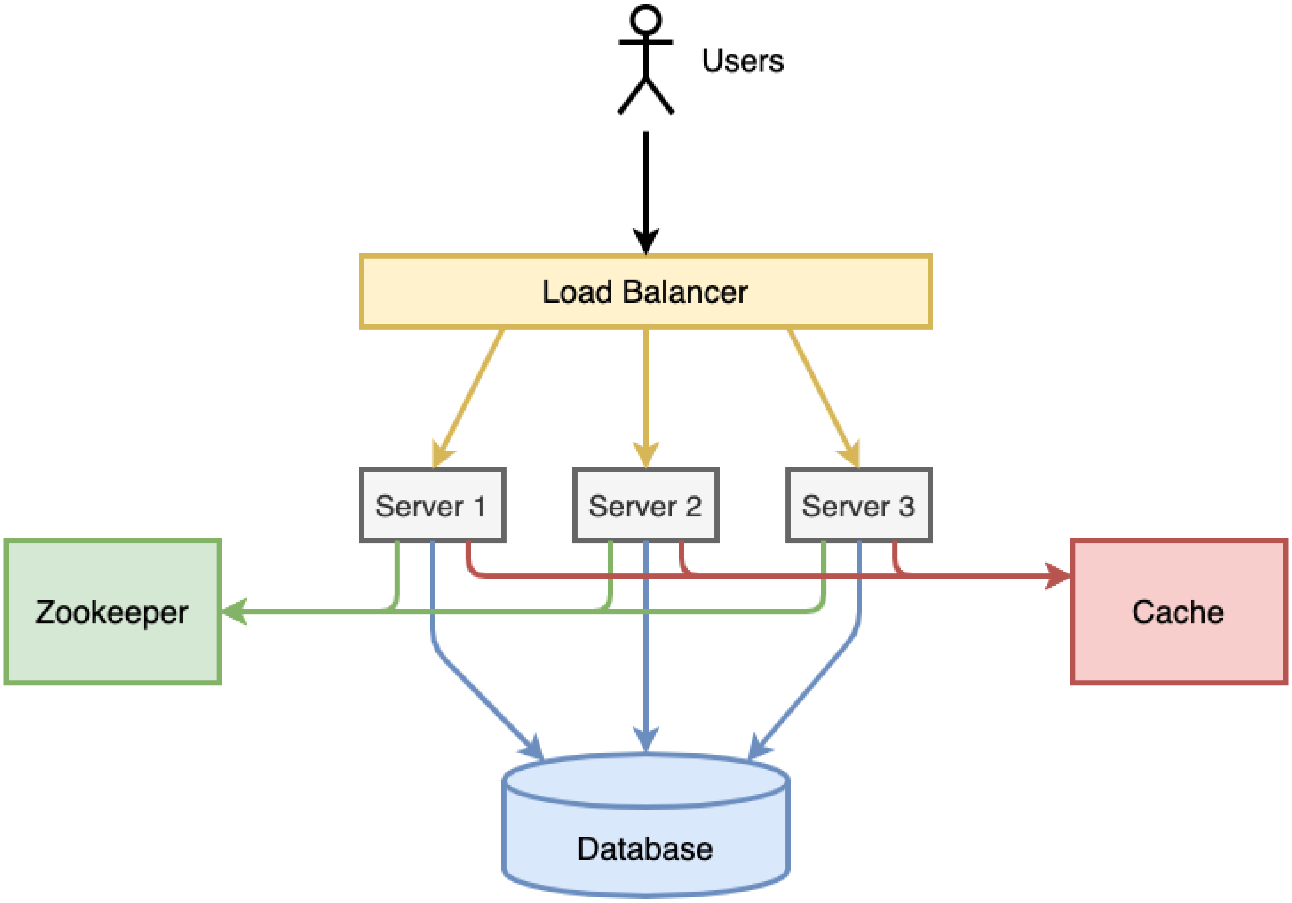
  - Table 1: URL
    - Description
      - Store URL mappings.
    - Columns

Column Name	Column Type	PK	Description
ID	int	PK	
ShortUrl	string		The short URL.
LongUrl	string		The original long URL.

- **Database**

  - NoSQL
    - Reason
      - No relation need to look up.
      - NoSQL is good at scaling.

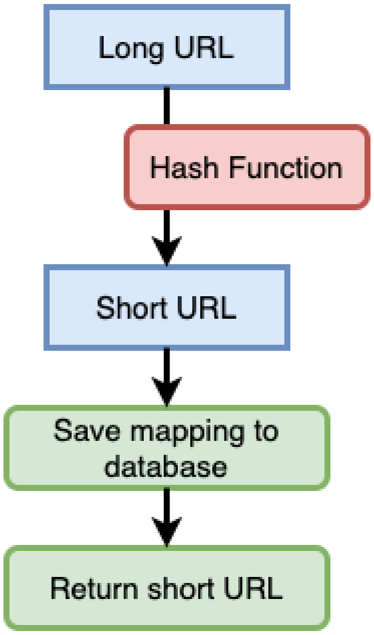
High-level design



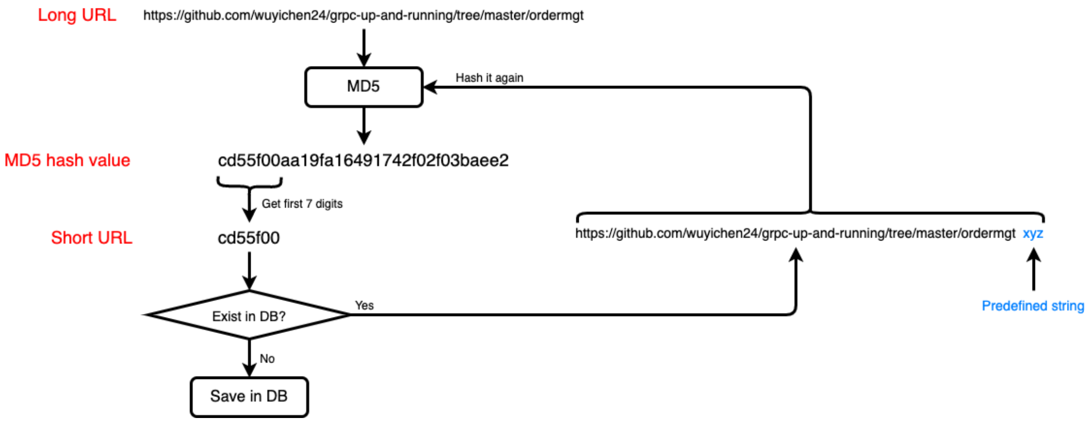
- Zookeeper
  - Distributed coordinator to give each server a unique unused range of keys.
- Cache
  - Stores the top 20% most used URLs.
  - When a server receives a URL query request, it can search the cache first. It the target URL is in in the cache, it can query the database.
- Database
  - Stores URLs and users.

Detailed design

- URL shortening
  - Process

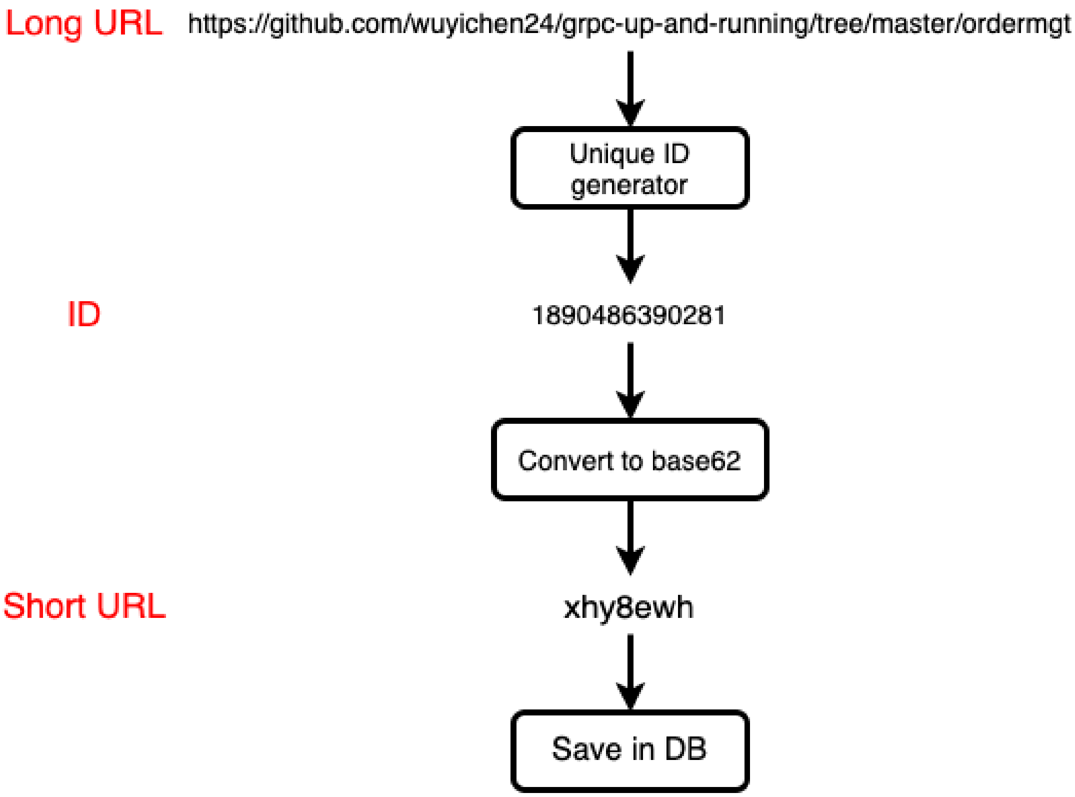


- Choices of hash function
    - Use existing hashing algorithm with collision resolution
        - A hash value from an existing hashing algorithm (CRC32, MD5, SHA-1, SHA-2, etc.) is too long, so we cannot use it directly. Our solution is to only use the first 7 characters of a hash value from an existing hashing algorithm.
        - Using the first 7 characters can result in a hash collision more easily.
        - If the first 7 characters has a hash collision, recursively append a new predefined string to the long URL and hash the new long URL again, unitl no hash collision.


Long URL
https://github.com/wuyichen24/grpc-up-and-running/tree/master/ordermgt
MD5
Hash it again
MD5 hash value
cd55f00aa19fa16491742f02f03baee2
Get first 7 digits
Short URL
cd55f00
https://github.com/wuyichen24/grpc-up-and-running/tree/master/ordermgt xyz
Predefined string
Exist in DB?
Yes
No
Save in DB

    - Use base62 conversion
        - Convert the unique ID (numeric value) of the new row for the URL mapping table from base 10 to base 62.
        - Example

        - The long URL is `https://github.com/wuyichen24/grpc-up-and-running/tree/master/ordermgt` .
        - The new ID for the new row is `1890486390281` .
        - Convert the ID from base 10 to base 62: `xhy8ewh` .
        - The short URL will be `https://abc.com/xhy8ewh` .



- **Uniqueness of short URLs**
    - Factors
        - Number of all possible characters in one digit.
        - The length of a short URL (Number of digits).
    - Calculation: Number of unique URLs = Number of all possible characters in one digit$^{\text{Number of digits}}$
    - Evaluation tradeoffs
        - Keep short URL as short as possible.
        - Don't let unique short URLs run out easily (Maximal number of URLs > Total number of short URLs created in 5 years).
    - Solutions

Number of all possible characters in one digit	Length of URLs	Maximal number of URLs
Only numbers (0-9) = 10	7	10^7 = 10 million

Number of all possible characters in one digit	Length of URLs	Maximal number of URLs
Base36 ([0-9, a-z]) = 36	7	36^7 = 78 billion
Base62 ([0-9, a-z, A-Z]) = 62	7	62^7 = 3.5 trillion

Key points

- Use Zookeeper as the distributed coordinator to solve the key conflict problem among multiple servers.

References

- https://www.educative.io/courses/grokking-the-system-design-interview/m2ygV4E81AR
- https://www.youtube.com/watch?v=eCLqmPBIEYs&t=1s&ab_channel=KAEducation
- https://www.scopulus.co.uk/tools/hexconverter.htm