Answers

1. How can we enhance the backend to have faster performance for frequently accessed “short URLs”?

We can create a table for "frequently accessed short URLs" that holds the key from the table "short_URLs" of the URLs that are frequently accessed, we can check which of the short URLs is frequently accessed by running a query on the tables "Short_URLs" and "Redirection" (each list in Short_URLs has his own Redirection Table), then after we know the amount of redirections happened from each short URL in the last day/hour/min (depends on how 'frequently accessed' is defined) we can know which short URLs are frequently accessed. We can run the query every day (for example) to update the 'frequently accessed' table. Once we have a table for the frequently accessed and a shortURL was fired, we will scan the 'frequently accessed' table and then the rest of the URLs in the 'Short URLs' table.

2. How can we enhance the implementation to have one “short URL” redirect to a few “long URLs”, based on redirect percentage? For example:
for http://localhost/sh54

a. 30% of the redirections will go to www.somethingverylong.com/1234.html

b. 70% of the redirections will go to www.somethingverylong.com/3456.html

to implement this requirement we need to change the table's structure. The Short_URLs table will hold only the key (short_URL), for each short_URL we will save another table called "Pos_Redirection" containing the percentage needed (30% will be 30, 70% will be 70, so on), and the long_URL (as text), each list in Short_URLs has his own Pos_Redirection table, then, in the function get_short_url we will use a random number between 0 to 1 (0 is 0% and 1 is 100%), then, regarding to the example, if the random number will be between 0.0 to 0.3 we will redirect to www.somethingverylong.com/1234.html, if the random number will be between 0.3 to 1 we will redirect to www.somethingverylong.com/3456.html, this will be possible as we now have for each short_URL its redirections saved along with a column that defines probability (the sum of this column must be 1). (the code needs to fit to the data stored in the possibility column, that way if we got 0.21 it will send a request to the Db for for the long_url with the possibility 30, if we got 0.65 it needs to know that for any random number higher the 0.3 and lower then 1 it needs to send a request to the Db for the long_url with the possibility 70)

3. How can we enhance the implementation to report the top 5 redirected URLs in the last hour?

Each list in Short_URLs has his own table of redirections, so we can create a new endpoint to the server that calculates how many redirections occurred in the last hour for each short_URL, after sorting the table we received in the query we can return only the top 5 lists.

4. How can we prevent an attack, in which a malicious client fills the database with millions of URL redirections?

We can set the server to only save lists in a difference of 1 second or a different period of time so that no attacker can make the server save so much lists in such a short time. Also, we can make the server save up to 10 (or a different number) URLs in an hour per IP.

5. What happens after the user presses the “Create short URL” button? Please detail as much as possible.

The User clicks the button, then an ajax POST request is created, containing the long_URL of the client, the call gets to gsurl (get short URL) endpoint in the server, then the server calculates the number of lists in the table Short_URLs (in order to make a unique id for the list, to save under the string 'key'), then a request to the Db has been created to insert a new list to Short_URLs table, the server closes the connection to the db and returns the new_url that's been created by the server and already saved, the client gets the response and changes the value of short_url parameter in the Vue.js's Object that specify the short_URL.