Prarthana Angadi

LogicMonitor MiniURL HackerRank Submission

- **Pros & Cons Design**

- Pros

1. Converting a long url to short url is a POST request, retrieving a short url from a long url is GET request and black listing a short url is PUT request to the web service.

2. 62base number conversion is used to convert a long counter variable starting from 0 to store records in the database.

3. Synchronized block is used to access the counter variable, so that no 2 threads read the same value, encode in 62 base number and try to insert a duplicate entry in the Database.

3. 6 bits out of 62 are used to represent a unique Short URL. These 6 bits are generated using [0-9a-zA-Z] characters only. Maximum 57 billion long URLs can be converted without any collision.

4. Data is sent & received in the JSON format.

5. Alternative designs which involve hashing, by using algorithms like md5, sha1 or using insertion timestamps result in alot of conflicts thereby reducing the overall efficiency of System.

6. Server side validations are in place to handle error situations

7. If a URL is already converted to a short URL, then the same short url is returned to the user. New entry is not created in the Database, inorder to avoid duplication.

8. If URL is already blacklisted, then user access is forbidden.

- **Cons**

1. Every valid request opens a new Database connection. Thereby increasing the load on the Database and also the response time is increased.

2. Short URL is currently the Primary Key in the Database. Current system does not handle the situation where the counter increases beyond 57 billion, which will result in collision and causing a Database access failure because duplicate entries cannot be created in the Database.

3. Custom Short URLs cannot be created.

- **Scalable Design**

1. Instead of opening a new Database connection everytime a valid request hits the Database, maintain a queue for the requests. Multiple threads are monitoring the queue. Also maintain a pool of available connections to access the Database. Whenever a new request arrives push in the queue. If any connection is available then the thread assigns the connection to the request else the request waits in the queue. Thereby reducing the load on the Database.

2. Inorder to minimize access to the Database, maintain cache which stores the most recently accessed URLs. Short URL to Long URL cache and vice versa.

3. Currently the system has only one MySQL machine but multiple machines should be used because if only one machine is being used, failure in that one machine may result in the failure of the entire system. Also to increase the throughput maintain more machines with Database copies.

4. When we have multiple machines we cannot use this global counter variable concept. We cannot store the counter variable on one machine because each incoming request would require an additional request to fetch the counter variable from that machine.

5. We divide the 57 billion short urls range into chunks and assign them to each machine in the network. Different locations use different web server and cache server. All the areas share a Database used to match the users to the closest web server (through DNS) when they have a miss on the cache.

6. Each machine has its own counter variable which has a particular range.

a. Hash the incoming long url using 62 base number, locate the machine. Once the machine is found check the cache server for an existing entry if found return the short url to the user else utilize the last used counter variable on that machine, increment it and hash it again, generate the short url & store in the Database.

b. For an incoming short url, decode the 62 base number into a decimal counter number, locate the machine. Once the machine is found check the cache server for an existing entry if found return the long url to the user else utilize the incoming short url as a key and fetch the entry from the database.

7. To minimize collisions once the counter reaches 57 billion we can maintain another column in the Database which is the popularity column, which maintains the number of times the longURL or the short URL is being accessed. Once collision is detected we can assign the short URL of the least popular long URL to this new request. Inorder to override a record we need to assign a validity period to the old long URL during the creation of its short URL and also inform the user of about the same. To determine a validity period the system must be analysed carefully, frequent access to a particular URL during peak times must be taken into consideration.