Design a URL Shortening

1. Why we use URL Shortening Service?
   1. This application helps us to save lots of space. To illustrate this, if we send mail or push notification for mobile phones, then we can save a lot of space. In addition hiding the actual URL is provided with this service.
   2. Notice that shortener url service is nearly 1/3 size of actual size.
2. Requirements and Goals of System
   1. There are much important requirements for URL shortener service.
   2. When user give the actual URL, this actual URL should save database and return the short URL to user
   3. Notice that short URL’s should be unique.
   4. Whenever user can enter the short URL, system should redirect user to actual URL.
   5. Notice that this short URL should have an expire time.
   6. Moreover some performance of system should be provided
      1. System should be highly reliable. This means that every actual URL should be saved and not deleted.
      2. System should have a minimum latency. This means that when user enter the short URL, user should quickly redirect the actual URL with minimum latency. For this purpose we use data sharding and caching.
      3. System should be highly available. This means that when user enter the short URL, he or she should reach the actual URL at the same time. For this purpose, we can use data replication.
   7. In addition user can save the short URL with custom alias. But for this property, we can restrict user with character limits.
   8. To monitor system, we can save the access time, access region, browser or platform.
   9. Finally, these short URL’s should not be predictable.
3. Capacity Estimation
   1. Let’s think about our system has 100 million users and each users enter the five new URL in a month. These means that every month, 500 million new URL comes.
   2. If we think about every actual URL size is 100 bytes (100 character represented) then in a month, 100 bytes \* 500 M = 50 GB new data comes. This means that in a 10 years later, 50 GB \* 12 \* 10 = 120 \* 50 GB = 6000 GB = approximately 6 TB data must saved.
   3. Notice that if the ratio of read and write operation is 10 : 1 then 500 million new URLS writes and 5 billion URLS reads.
      1. Traffic estimates
         1. In a second = 500 million / (30\*24\*3600) = ~200 URL / s to write
         2. In a second = ~200 \* 10 = 2K URL / s to read
      2. Bandwith estimates
         1. In a second = 200 URL \* 100 bytes = 20 KB / s to write
         2. In a second = 2 K URL \* 100 bytes = 200 KB / s to read
      3. Caching estimate
         1. Notice that if we realize that %20 rule of memory.
            1. In a day = 2 K \* 3600 \* 24 \* 0.2 = ~35 GB
4. System API’s
   1. We can use REST API to connect between system and devices. There are two main APIs that are createURL and deleteURL. All of these have develop\_api\_key for controlling the usage and limit of using API.
      1. createURL(key, custom\_alias=None, user\_name=None, expire\_date = None) Notice that this returns string short URL
      2. deleteURL(key, short\_url) return http api response.
5. Database Design
   1. For this software we can use NoSql like as DynamoDB, MongoDB or Cassandra. Amazon provides Dynamo databases for this purpose. Why we can use NoSql is easy. We have lots of URL and any URL has no relation to other URLs.
   2. For this software we basically have three tables. One is URL table, second is User table and third is URL and User relation table. Notice that NoSql has no relationship between tables and NoSql has no support ACID properties. This basically work with CAP Theorem.
6. Basic System Design
   1. First we think about how can we provide actual unique short URL and how can we format actual URL. The simpliest way to encode URL ise use MD5 or SHA256 hashing with base36 or base62 or base64 hashing. So we think that 8 unique letters to use hashing and base64 is used, then approximately 280 trillion possible strings are created and this number is big enough for unique URLs.
      1. Think basic:
         1. Client enter the actual URL
         2. Short URL goes to server
         3. Server encode the URL and try to save database
         4. If database has this encode URL return fail and want to new short URL to server
         5. If database has no this short URL and it saved database and server returns short URL
      2. But how we can handle unique key operations
         1. For this purpose we think that user can be online or offline. So we can use key generation service for this purpose.
         2. Key generation service has two database and one for keeping not used key and another for keeping used key.
         3. Notice that key generation service created unique keys before and this provides minimizing latency.
         4. If we have one more server then we must synchronize the key to eliminate obtaining same key on concurrent time.
7. Data Partitioning and Replication
   1. Billions of URL should be saved and because of that we need to data partition. We can prefer two types of partition. First one is horizontal partition and second one is vertical partition. For this software we use horizontal partition.
   2. We can use hash-based partitioning. With this property, our hashing randomly distribute URLs into different partitions.
   3. For replication we can replicate all servers at least 2 another copy services.
8. Cache
   1. We can use Amazon elastiCache for caching operations.
   2. LRU caching mechanism uses for this purpose
   3. Notice that cache data is a mirror copy of actual data. Each copy data should have an expire date.
9. Load Balancer
   1. Each server layers can have load balancer.
      1. Clients and application servers
      2. Application servers and database servers
      3. Application servers and cache servers
   2. We can use round robin but this has some problems so we can use intelligent LB approach.
10. More Visually
    1. Clients enter the url
    2. Proxy server take the requests
    3. Proxy server looks the request whether its cache has this short url or not.
    4. If not then redirect the request to suitable application server by load balancer.
    5. If the request is related new URL, application servers contact with Key generation service.(key-DB)
    6. Application server redirect the request cache servers and database servers by load balancers.