System Design (/courses/system-design)   ⊡ Show Notes
/ System Design Interview Questions (/courses/system-design/topics/interview-questions/)
/ Design Url Shortener

Design URL Shortener   [Bookmark]

● ◀   Design a URL shortening service, like bit.ly

https://www.interviewbit.com/courses/programming   SHORTEN

Blog (https://blog.interviewbit.com)   About Us (/pages/about_us/)   FAQ (/pages/faq/) |

Contact Us (/pages/contact_us/) | Terms (/pages/terms/) | Privacy Policy (/pages/privacy/)

Programming — InterviewBit
interviewbit.com/courses/programming/

System Design Interview Questions (/courses/system-design/) |

http://bit.ly/2bRaM86 [COPY] Google Interview Questions (/google-interview-questions/) | 0 ılıllıı

Facebook Interview Questions (/facebook-interview-questions/) |

Amazon Interview Questions (/amazon-interview-questions/) |

Microsoft Interview Questions (/microsoft-interview-questions/) | Puzzles Questions (/puzzles/)

● **Features:**

f Like Us (https://www.facebook.com/interviewbit)   🐦 Follow Us (https://twitter.com/interview_bit)

" *This is the first part of any system design interview, coming up with the*
✉ Email (mailto:hello@interviewbit.com)
features which the system should support. As an interviewee, you should try
to list down all the features you can think of which our system should support.
Try to spend around 2 minutes for this section in the interview. You can use
the notes section alongside to remember what you wrote. "

Shortening: Take a url and return a much shorter url.

Redirection: Take a short url and redirect to the original url.
Ex: http://goo.gl/GUKA8w => http://www.interviewbit.com/courses/programming/topics/time-complexity/

Custom url: Allow the users to pick custom shortened url.
Ex: http://www.interviewbit.com/courses/programming/topics/time-complexity/ => http://goo.gl/ib-time

Analytics: Usage statistics for site owner.
Ex: How many people clicked the shortened url in the last day?
Gotcha: What if two people try to shorten the same URL?

● **Estimation:**

> ❝ *This is usually the second part of a design interview, coming up with the estimated numbers of how scalable our system should be. Important parameters to remember for this section is the number of queries per second and the data which the system will be required to handle.*
> *Try to spend around 5 minutes for this section in the interview.* ❞

❓ ◀ **Q:** How many queries per second should the system handle?(Assuming 100 Million new URLs added each month)

Hint: Assuming average lifetime of a shortened URL is 2 weeks and 20% of websites creates 80% of the traffic, we see that we'll receive around 1 Billion queries in a month.

A: 400 queries per second in total. 360 read queries and 40 write queries per second.

❷ ◀ **Q:** How much data will we need to store so that we don't have to restructure our architecture for the next 5 years considering constant growth rate?

Q: How many URLs will we need to handle in the next 5 years?

💬 ()

Hint: Earlier we saw, we would see 100 Million new URLs each month. Assuming same growth rate for next 5 years, total URLs we will need to shorten will be 100 Million * 12 * 5 = 6 Billion.
A: 6 Billion.

Q: What is the minimum length of shortened url to represent 6 Billion URLs?

💬 ()

Hint: We will use (a-z, A-Z, 0-9) to encode our URLs. If we call x as minimum number of characters to represent 6 Billion total URLs, then will be the smallest integer such that x^62 > 6*10^9.
A: Log $(6*10^9)$ to the base 62 = 6

A: 3 TeraBytes for URLs and 36 GigaBytes for shortened URLs
Note that for the shortened URL, we will only store the slug(6 chars) and compute the actual URL on the fly.

💬 ()

❷ ◀ **Q:** Data read/written each second?

Hint: Data flow for each request would contain a shortened URL and the original URL. Shortened URL as we saw earlier, will take 6 bytes whereas the original URL can be assumed to take atmost 500 bytes.

A: Written : 40 * (500 + 6) bytes, Read : 360 * (500 + 6) bytes

💬 ()

● **Design Goals:**

> ❝
> **Latency** - Is this problem very latency sensitive (Or in other words, Are requests with high latency and a failing request, equally bad?). For example, search typeahead suggestions are useless if they take more than a second.
> **Consistency** - Does this problem require tight consistency? Or is it okay if things are eventually consistent?
> **Availability** - Does this problem require 100% availability?
>
> *There could be more goals depending on the problem. It's possible that all parameters might be important, and some of them might conflict. In that case, you'd need to prioritize one over the other.* ❞

❷ ◀ **Q:** Is Latency a very important metric for us?

A: Yes. Our system is similar to DNS resolution, higher latency on URL shortener is as good as a failure to resolve.

💬 ()

❷ ◀ **Q:** Should we choose Consistency or Availability for our service?

A: This is a tricky one. Both are extremely important. However, CAP theorem dictates that we choose one. Do we want a system that always answers correctly but is not available sometimes? Or else, do we want a system which is always available but can sometime say that a URL does not exists even if it does?
This tradeoff is a product decision around what we are trying to optimize. Let's say, we go with consistency here.

💬 ()

❷ ◀ **Q:** Try to list down other design goals?

A: URL shortener by definition needs to be as short as possible. Shorter the shortened URL, better it compares to competition.

💬 ()

● **Skeleton of the design:**

> ❝ *The next step in most cases is to come up with the barebone design of your system, both in terms of API and the overall workflow of a read and write request. Workflow of read/write request here refers to specifying the important components and how they interact. Try to spend around 5 minutes for this section in the interview.*
>
> ***Important*** *: Try to gather feedback from the interviewer here to indicate if you are headed in the right direction.* ❞

❷ ◀ **Q:** How should we define our APIs?

A:

```
ShorteningAPI(url) {store the url_mapping and return hash(url)}

RedirectionAPI(hash) {redirect_to url_mapping[hash]}
```

Both APIs are very lightweight, their computation will not the bottleneck plus we don't have to store any session information about users.
Basically, we are trying to build a service which serves as a huge HashMap

💬 ()

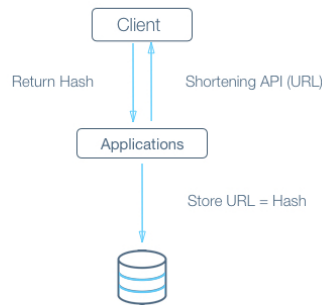❷ ◀ **Q:** How would a typical write query look like?

A: Components:

Client (Mobile app / Browser etc) which calls ShorteningAPI(url)
Application server which interprets the API call and generates the shortened
hash for the url

Database server which stores the hash => url mapping

HIGH LEVEL DESIGN (WRITE)

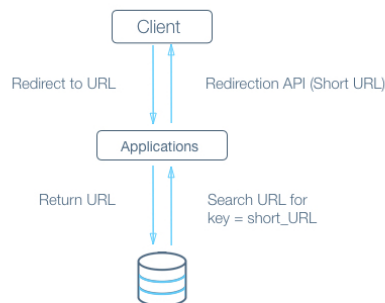**Q:** How would a typical read query look like?

A: Components:

Client (Mobile app / Browser, etc) which calls RedirectionAPI(short_url)
Application server which interprets the API call, extracts the hash from short_url,
asks database server for url corresponding to hash and returns the url.
Database server which stores the hash => url mapping

HIGH LEVEL DESIGN

● **Deep Dive:**

> " *Lets dig deeper into every component one by one. Discussion for this section will take majority of the interview time(20-30 minutes).* "

❓ ◀ **Q:** How should we compute the hash of a URL?

> *Gotcha:* How should we handle the case where two separate URL gets shortened to the same URL?
>
> 💬 ()
>
> **A:** We can use a list of salts in case of collision.
> For each read request, we can compute all possible shortened URLs using our list of salts and query for them in parallel to save time.

A: convert_to_base_62(md5(original_url + salt))[:6](first six characters)
Links: MD5-Wiki (https://en.wikipedia.org/wiki/MD5) Base 62 Conversion-Stackoverflow (http://stackoverflow.com/questions/1119722/base-62-conversion-in-python)

Gotchas:

Directly encoding URL to base_62 will allow a user to check if a URL has been shortened already or not, reverse enginnering can lead the user to the exact hash function used, this should not be allowed. Therefore randomization has to be introduced. Also, if two users shortens the same URL, it should result in two separate shortened URL(for analytics)

Database ID encoded to base_62 also won't be suitable for a production environment because it will leak information about the database. For example, patterns can be learnt to compute the growth rate(new URLs added per day) and in the worst case, copy the whole database.

**❓ ◀ Q:** How would you take care of application layer fault tolerance?

Q: How do we handle the case where our application server dies?

💬 ()

A: The simplest thing that could be done here is to have multiple application server. They do not store any data (stateless) and all of them behave the exact same way when up. So, if one of them goes down, we still have other application servers who would keep the site running.

HIGH LEVEL DESIGN
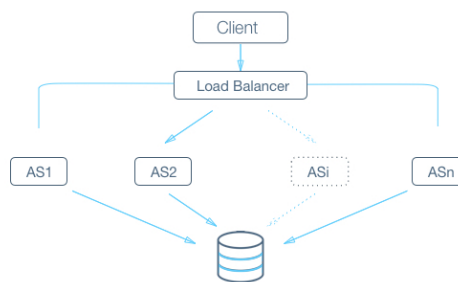
Q: How does our client know which application servers to talk to. How does it know which application servers have gone down and which ones are still working?

💬 ()

A: We introduce load balancers. Load balancers are a set of machines (an order of magnitude lower in number) which track the set of application servers which are active (not gone down). Client can send request to any of the load balancers who then forward the request to one of the working application servers randomly.

A: If we have only one application server machine, our whole service would become unavailable. Machines will fail and so will network. So, we need to plan for those events. Multiple application server machines along with load balancer

is the way to go.

💬 ()

❓ ◀ **Q:** What all data should we store?

A: Data storage layer: Hash => URL mapping.
Billions of small sized(1kb) object. There is no relationship between objects.
We would also need to store data for analytics, for example, how many times
was the url opened in the last hour?

💬 ()

❓ ◀ **Q:** Should we choose RDBMS or NOSQL?

Hint : Things to consider :
Are joins required :
NoSQL databases are inefficient for joins.
In this case, assuming we don't need analytics, we only need to answer the query
"Given a hash, give me the corresponding URL" - a standard key to value lookup. As
such, we don't need any joins here.
Size of the DB :
If the size of the data is so small that it fits on a single machine's main memory, SQL
is a clear winner. SQL on a single machine has next to zero maintenance overhead
and has great performance with right index built. If your index can fit into RAM, its
best to go with a SQL solution. Lets analyze our current case :

Assumptions :
# of writes per month: 100 Million (Refer to estimations section)
Avg size of URL : 500 bytes
Provisioning for : 5 years
Space required : 500 bytes * 100M * 12 * 5 = 3TB

3TB of data can fit on a single machine's hard disk. However, the index might not. If
we store all data on a single machine, our write and read operations would be very
slow (Page swaps for indices). Given read latency is critical for us, we can't store the
data on a single machine.
So, a SQL solution will have a sharding overhead. Most NoSQL solutions however
are built with the assumption that the data does not fit on a single machine and

hence have sharding builtin.

A: As discussed in hints, NoSQL would be a better fit for our case.
Since we are optimizing for consistency over availability, we will choose a
NoSQL DB which is highly consistent like HBase.

💬 ()

❷ ◀ **Q:** What would the database schema look like?

A: We want to store the mapping from hash -> URL which is ideal for a key
value store. In NoSQL domain, we need to be careful when designing the key as
entries are indexed by the key.
In our case, since we will only query by hash which will never update, our key
will be hash with the value being the URL.

💬 ()

❷ ◀ **Q:** How would we do sharding?

A: HBase inherently use consistent hashing for sharding. We explain it in detail
at https://www.interviewbit.com/problems/sharding-a-database/
(https://www.interviewbit.com/problems/sharding-a-database/). Since our total
data size is a few TBs, we don't need to think about sharding across datacentres
either.

💬 ()

❷ ◀ **Q:** How would we handle a DB machine going down?

A: NoSQL again has single machine failure handling inbuilt. A NoSQL DB like
HBase would have an availability issue for a few seconds for the affected keys,
as it tries to maintain tight consistency.

💬 ()

● **Bonus Exercise :**

❓ ◀ **Q:** How can we optimize read queries?

A: Caching can be used to reduce average read time. Since most URL's are accessed for only a small amount of time after its creation, LRU cache fits our use case. Also, if a shortened URL goes viral, serving it through a cache will reduce the chances overloading the database. Redis and Memcached are examples of well known and widely used caches.
Redis-Tutorial (http://try.redis.io/) is an interactive tutorial for Redis.
Redis vs Memcached (http://stackoverflow.com/questions/10558465 /memcached-vs-redis) lists down comparison between them.

💬 ()

❓ ◀ **Q:** How would you shard the data if you were working with SQL DB?

Hint: Consistent Hashing - Stackoverflow (http://stackoverflow.com/questions /144360/simple-basic-explanation-of-a-distributed-hash-table-dht)

A: We can use integer encoding of the shortened URL to distribute data among our DB shards.
Assuming we assign values from 0 to 61 to characters a to z, A to Z, and 0 to 9, we can compute the integer encoding of the shortened URL.
We can see that the maximum value of integer encoding will be less than 10^{13}, which we can divide among our DB shards.
We will use consistent hashing to ensure we don't have to rehash all the data again if we add new DB shards later.

💬 ()

❓ ◀ **Q:** How would you handle machine dying in the case of SQL DB?

A: This is a bit tricky. Obviously, for every shard, we need to have more than one machine
We can have a scheme better known as master slave scheme, wherein there is one machine(master) which processes all writes and there is a slave machine which just subscribes to all of the writes and keeps updating itself. In the event that the master goes down, the slave can take over and start responding to the

read queries.

💬 ()

✓◀   🏆 **You have now mastered this problem!**

Discussion

S

SaurabhGarg09   🕐 6 months ago

**How to deal with duplicate url ? In this case system will
generate a new show url**   1

Reply (https://discuss.interviewbit.com/session/sso?return_path=https:
//discuss.interviewbit.com/t/how-to-deal-with-duplicate-url-in-this-case-system-will-
generate-a-new-show-url)

A

anshul_jain   🕐 8 months ago

**Isn't the index size of total 3TB <hash key, URL> data
36GB?**   1

Reply (https://discuss.interviewbit.com/session/sso?return_path=https:
//discuss.interviewbit.com/t/isnt-the-index-size-of-total-3tb-lt-hash-key-url-gt-
data-36gb)

💡 **Got suggestions ? We would love to hear your
feedback.**   ♥ **Loved InterviewBit ? Write us a testimonial.
(http://www.quora.com/What-is-your-review-of-
InterviewBit)**

R

roy-zhang   🕐 6 months ago

Should I return the same tiny URL for two users who try to shorten the same long URL? 1

Reply (https://discuss.interviewbit.com/session/sso?return_path=https: //discuss.interviewbit.com/t/should-i-return-the-same-tiny-url-for-two-users-who-try- to-shorten-the-same-long-url)

Y

yashveer-rana   🕐 8 months ago

X^62 > 610^9 should be 62^x > 610^9 1

Reply (https://discuss.interviewbit.com/session/sso?return_path=https: //discuss.interviewbit.com/t/x-62-gt-610-9-should-be-62-x-gt-610-9)

G

Gamechanger   🕐 4 months ago

Why to convert to base_62 and not base_70 1

Reply (https://discuss.interviewbit.com/session/sso?return_path=https: //discuss.interviewbit.com/t/why-to-convert-to-base-62-and-not-base-70)

P

preeti_khurana   🕐 about 2 years ago

Question: How many URLs will we need to handle in the next 5 years? 1 Hint: Earli 0

Reply (https://discuss.interviewbit.com/session/sso?return_path=https: //discuss.interviewbit.com/t/question-how-many-urls-will-we-need-to-handle-in-the- next-5-years-1-hint-earli)

A

agniswar_bakshi   🕐 almost 2 years ago

Do the tiny urls need to have an expiry time? 0

Reply (https://discuss.interviewbit.com/session/sso?return_path=https: //discuss.interviewbit.com/t/do-the-tiny-urls-need-to-have-an-expiry-time)

A

ahmedjamalmaaz   🕐 7 months ago

How about using a BloomFilter to make write faster?

Reply (https://discuss.interviewbit.com/session/sso?return_path=https: //discuss.interviewbit.com/t/how-about-using-a-bloomfilter-to-make-write-faster)

G

gauravraval · almost 2 years ago

QPS:100 million / month = 3 million / day (100m/30) = 125000/hour (3m/24) = 2083   0

Reply (https://discuss.interviewbit.com/session/sso?return_path=https://discuss.interviewbit.com/t/qps-100-million-month-3-million-day-100m-30-125000-hour-3m-24-2083)

S

santosh · 2 months ago

How does the database handle expired URLs? e.g. just replace the key's value with TinyUrl's 404 page?   1

Reply (https://discuss.interviewbit.com/session/sso?return_path=https://discuss.interviewbit.com/t/how-does-the-database-handle-expired-urls-e-g-just-replace-the-keys-value-with-tinyurls-404-page)

A

abhishek_dhadse · 4 months ago

Do the figures matter from interview perspective?   0

Reply (https://discuss.interviewbit.com/session/sso?return_path=https://discuss.interviewbit.com/t/do-the-figures-matter-from-interview-perspective)

Got suggestions ? We would love to hear your feedback.

Loved InterviewBit ? Write us a testimonial. (http://www.quora.com/What-is-your-review-of-InterviewBit)