

[首页](#)[话题](#)[发现](#)[消息](#)[提问](#)

李新存

[短链接](#)[短网址](#)[短域名](#)[URL 设计](#)[修改](#)

短 URL 系统是怎么设计的？ [修改](#)

新浪微博那种输入完全url转换为短url是怎么设计的 [修改](#)

[添加评论](#) [分享](#) · [邀请回答](#)[举报](#)

6 个回答

[默认排序](#)

iammutex，彩石手机CTO - 做最好的中老年智能手机



544 人赞同

看了一下 @Tang minyi 的回答，也非常不错，如果面试者这么回答，我也会很高兴的。
对于我用词比较激烈的问题，我面试时真实想法既是如此，也难得润色了。

这个问题看到就想答。

个人相关：三年前在公司做过一个短地址服务，目前在线上跑。

而这个问题，也是我现在招聘面试题里面必考的一道，这一道题里面有很多可考的地方，能够相对综合的考察候选人的功力。

最烂的回答

实现一个算法，将长地址转成短地址。实现长和短一一对应。然后再实现它的逆运算，将短地址还能换算回长地址。

这个回答看起来挺完美的，然后候选人也会说现在时间比较短，如果给我时间我去找这个算法就解决问题了。但是稍微有点计算机或者信息论常识的人就能发现，这个算法就跟永动机一样，是永远不可能找到的。即使我们定义短地址是100位。那么它的变化是62的100次方。 $62=10\text{数字}+26\text{大写字母}+26\text{小写字母}$ 。无论这个数多么大，他也不可能大过世界上可能存在的长地址。所以实现一一对应，本身就是不可能的。

再换一个说法来反驳，如果真有这么一个算法和逆运算，那么基本上现在的压缩软件都可以歇菜了，而世界上所有的信息，都可以压缩到100个字符。这~可能吗。

另一个很烂的回答

和上面一样，也找一个算法，把长地址转成短地址，但是不存在逆运算。我们需要把短对长的关系存到DB中，在通过短查长时，需要查DB。

怎么说呢，没有改变本质，如果真有这么一个算法，那必然是会出现碰撞的，也就是多个长地址转成了同一个短地址。因为我们无法预知会输入什么样的长地址到这个系统中，所以不可能实现这样一个绝对不碰撞的hash函数。

比较烂的回答

那我们用一个hash算法，我承认它会碰撞，碰撞后我再在后面加1，2，3不就行了。

ok，这样的话，当通过这个hash算法算出来之后，可能我们会需要做btree式的大于小于或者like查找到能知道现在应该在后面加1，2，或3，这个也可能由于输入的长地址集的不确定性。导致生成短地址时间的不确定性。同样烂的回答还有随机生成一个短地址，去查找是否用过，用过就再随机，如此往复，直到随机到一个没用过的短地址。

正确的原理

上面是几种典型的错误回答，下面咱们直接说正确的原理。

正确的原理就是通过发号策略，给每一个过来的长地址，发一个号即可，小型系统直接用mysql的自增索引就搞定了。如果是大型应用，可以考虑各种分布式key-value系统做发号器。不停的自增就行了。第一个使用这个服务的人得到的短地址是xx.xx/0 第二个是 xx.xx/1 第11个是 xx.xx/a 第依次往后，相当于实现了一个62进制的自增字段即可。

几个子问题

1. 62进制如何用数据库或者KV存储来做？

其实我们并不需要在存储中用62进制，用10进制就好了。比如第10000个长地址，我们给它的短地址对应的编号是9999，我们通过存储自增拿到9999后，再做一个10进制到62进制的转换，62进制数即可。这个10~62进制转换，你完全都可以自己实现。

2. 如何保证同一个长地址，每次转出来都是一样的短地址

上面的发号原理中，是不判断长地址是否已经转过。也就是说用拿着百度首页地址来转，我给一个xx.xx/abc 过一段时间你再来转，我还会给你一个 xx.xx/xyz 。这看起来挺不好的，但是不好在哪里呢？不好在不是——对应，而一长对多短。这与我们完美主义的基因不符合，那么除此以外还有什么不对的地方？

有人说它浪费空间，这是对的。同一个长地址，产生多条短地址记录，这明显是浪费空间的。那么我们如何避免空间浪费，有人非常迅速的回答我，建立一个长对短的KV存储即可。嗯，听起来有理，但是。。。这个KV存储本身就是浪费大量空间。所以我们是在用空间换空间，而且貌似是在用大空间换小空间。真的划算吗？这个问题要考虑一下。当然，也不是没有办法解决，我们做不到真正的一一对应，那么打个折扣是不是可以搞定？这个问题的答案太多种，各有各招，我这就可不说了。（由于实在太多人纠结这个问题，请见我最下方的更新）

3. 如何保证发号器的大并发高可用

上面设计看起来有一个单点，那就是发号器。如果做成分布式的，那么多节点要保持同步加1，多点同时写入，这个嘛，以CAP理论看，是不可能真正做到的。其实这个问题的解决非常简单，我们可以退一步考虑，我们是否可以实现两个发号器，一个发单号，一个发双号，这样就变单点为多点了？依次类推，我们可以实现1000个逻辑发号器，分别发尾号为0到999的号。每发一个号，每个发号器加1000，而不是加1。这些发号器独立工作，互不干扰即可。而且在实现上，也可以先是逻辑的，真的压力变大了，再拆分成独立的物理机器单元。1000个节点，估计对人类来说应该够用了。如果你真的还想更多，理论上也是可以的。

4. 具体存储如何选择

这个问题就不展开说了，各有各道，主要考察一下对存储的理解。对缓存原理的理解，和对市面上DB、Cache系统可用性，并发能力，一致性等方面的理解。

5. 跳转用301还是302

这也是一个有意思的话题。首先当然考察一个候选人对301和302的理解。浏览器缓存机制的理解。然后是考察他的业务经验。301是永久重定向，302是临时重定向。短地址一经生成就不会变化，所以用301是符合http语义的。同时对服务器压力也会有一定减少。

但是如果使用了301，我们就无法统计到短地址被点击的次数了。而这个点击次数是一个非常意思的大数据分析数据源。能够分析出的东西非常非常多。所以选择302虽然会增加服务器压力，但是我想是一个更好的选择。

大概就是这样。

-----五一假期后更新-----

就回答一点大家最纠结的问题吧，就是如何实现同一个长地址多次转换，出来还是同一个短地址。

我上面其实讲到了，这个方案最简单的是建立一个长对短的hashtable，这样相当于用空间来换空间，同时换取一个设计上的优雅（真正的一对一）。

实际情况是有很多性价比高的打折方案可以用，这个方案设计因人而异了。那我就说一下我的方案吧。

我的方案是：用key-value存储，保存“最近”生成的长对短的一个对应关系。⁵⁴⁴ 是“最近”，也就是说，我并不保存全量的长对短的关系，而只保存最近的。比如采用一小时过期的机制来实现LRU。

这样的话，长转短的流程变成这样：

- 1 在这个“最近”表中查看一下，看长地址有没有对应的短地址
 - 1.1 有就直接返回，并且将这个key-value对的过期时间再延长成一小时
 - 1.2 如果没有，就通过发号器生成一个短地址，并且将这个“最近”表中，过期时间为1小时

所以当一个地址被频繁使用，那么它会一直在这个key-value表中，总能返回当初生成那个短地址，不会出现重复的问题。如果它使用并不频繁，那么长对短的key会过期，LRU机制自动就会淘汰掉它。

当然，这不能保证100%的同一个长地址一定能转出同一个短地址，比如你拿一个生僻的url，每间隔1小时来转一次，你会得到不同的短地址。但是这真的有关系吗？

编辑于 2015-06-16 83 条评论 感谢 分享 收藏 · 没有帮助 · 举报 · 作者保留权利

▲ **Wizmann**， 智情双低留隐患

13

13 人赞同

- ▼
1. 不要幻想使用压缩算法，对于URL这种不超过100bytes的字符串，压缩算法的压缩比通常都大于1。
 2. 不要幻想使用Hash映射，因为Hash冲突是不可控的。当然，我们有解决Hash冲突的N种方法，但是这只会增加系统的复杂度。

其它观点和iammutex基本一致。在整数（大整数）空间内分层，每层使用一个auto increment的primary key，使用id->url的映射来进行存储与查找。

后台挂个redis就好了。（当然也可以造轮子

==

Update：

亲测，64Bit的Url签名的冲突率非常低（Murmur64），并非不可以应用在系统中。

编辑于 2015-08-20 5 条评论 感谢 分享 收藏 · 没有帮助 · 举报 · 作者保留权利

▲ 匿名用户

7

7 人赞同

刚好本人负责短链接。我简单说下。

既然是短链接，那么地址池是有限的。所以在业务上会要求接受短链接失效，保证可回收利用。但是总有一些场景要求长期有效，所以可预留一些特殊的。

以上是最基本的系统功能要求。

至于技术上怎么做，太简单了。DB索引，随机都可以，搞个分库分表，读多写少。随机的时候碰撞了就重试几次，没什么问题。在对总短链接数做个监控，地址用的太多会造成随机碰撞越来越多，也可能是回收机制出问题了。

另外防攻击做好。别个把你链接池地址耗光。

匿了，不想被发现

发布于 2015-05-02 3 条评论 感谢 分享 收藏 · 没有帮助 · 举报 · 作者保留权利

544

▲ 玄魂工作室·玄魂，程序员/演示文档技术研发/业余写手

0 id自增，简单，实用

▼

发布于 2016-03-29 添加评论 感谢 分享 收藏 · 没有帮助 · 举报 · 申请转载

▲ 吹风，懒人一个

0

多个前端连接一个redis做全局id生成，实际试试，能抗住的的压力比想象的高。

▼ id空间的问题，根本不是问题，一个64位的有符号整数空间大的吓人。开金手指让它不够，把单个key换一组key。

大并发下，本身面对的问题还是一般网站面对的那些问题。

发布于 2015-05-03 添加评论 感谢 分享 收藏 · 没有帮助 · 举报 · 作者保留权利

▲ 王川，AWS老司机，球盲

3

3 人赞同

▼ 如果这是道面试题，我会推荐使用YOURLS，已有现成和完善的解决方案，重复造轮子的事真心意义不大。短域名的目的是啥？

1、缩短URL满足字数限制。

2、美观。

3、统计需要。

在这基础上，要求能够满足高并发。设计要点：

1、随机生成短域名：

```
$possible = "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";
```

```
$str = substr( str_shuffle( $possible ), 0, $length );
```

2、高效的短地址系统应采用nosql数据库中key-value型。key即为短地址字符串，value中至少两个字段：原地址和访问量，要求访问量能够实现原子递增功能。

编辑于 2016-09-13 添加评论 感谢 分享 收藏 · 没有帮助 · 举报 · 作者保留权利

4 个回答被折叠（为什么？）

李新存

填写话题经验，提升回答可信度



写回答...

544

☐ 匿名 |

允许规范转载 ▼

发布回答

关注问题

1124 人关注该问题



相关问题

- 为什么越来越多的网站域名不加「www」前缀？ 57 个回答
- 短网址有什么用？ 16 个回答
- 人人网，微博等网站在分享url的时候都会转换成短链接，这样有什么好处？ 15 个回答
- 短链接、短网址使用的是什么算法？ 6 个回答
- 国内有哪些靠谱的短链接服务？ 14 个回答

问题状态

最近活动于 2016-10-21 · [查看问题日志](#)
被浏览 51954 次，相关话题关注者 177 人

▲
544
▼