

设计思路

笔记本： tech

创建时间： 2022/4/21 13:09

更新时间： 2022/4/21 13:51

作者： 182wrras537

设计思路

1.需求分析

1. 首先理解需求的目的是，其实就是长链接与短链接之间的转换问题。用户输入长链接，系统给之生成短链接，方便传输，并且可以隐藏链接的内部信息，防止恶意用户猜测。参考实际业务中，
2. 可以得知，短链接和长链接是多对一的关系，即多个短链接可以对应一个长链接，但是一个长链接只能对应一个短链接，不然用户拿到这个短链接，不知道如何处理。

2.代码设计

对于长链接和短链接之前的存储和转换，其实设计好二者之间的映射关系即可。而且需求中说明映射关系存在jvm中即可，所以很容易的就想到了java中的map可以实现这种映射关系。由于代码过于简单，这里就不做架构设计图了。

对于接口一：接受长域名信息，返回短域名信息，只需要对map做put存储即可

对于接口二：接受短域名信息，返回长域名信息, 只需要对map进行取值即可

细节：

1.如果考虑到并发安全问题，这里的map选取最好选ConcurrentHashMap，保证线程安全问题

2.接口二，获取长域名，对map的存取，我们知道map的get方法，根据map的数据结果我们知道时间复杂度是O(1)，即根据hash算法，一次即可定位到。如果碰到hash冲突，只需要遍历桶上的链表或者红黑树，效率也会很高

3.真实业务场景下，如果考虑高并发的话，存在jvm中其实是不能满足该业务需求的。

原因：

a.服务采用springboot技术，我们知道springboot底层的web服务器是tomcat，而

c.单机下jvm存储，如果机器重启了，之前建立的映射数据关系都会清空。

互斥锁

```
sequenceDiagram
    participant T1 as 线程1
    participant T2 as 线程2
    T1->>L: 1. 查询缓存, 未命中
    T1->>L: 2. 获取互斥锁成功
    T1->>L: 3. 查询数据库重建缓存数据
    T1->>L: 4. 写入缓存
    T1->>L: 5. 释放锁
    T2->>L: 1. 查询缓存, 未命中
    T2->>L: 2. 获取互斥锁失败
    T2->>L: 3. 休眠一会儿, 再重试
    T2->>L: 4. 重试
    T2->>L: 5. 缓存命中
```

线程1

1. 查询缓存, 未命中
2. 获取互斥锁成功
3. 查询数据库重建缓存数据
4. 写入缓存
5. 释放锁

线程2

1. 查询缓存, 未命中
2. 获取互斥锁失败
3. 休眠一会儿, 再重试
4. 重试
5. 缓存命中

[illegible]

发现写入时间在 376ms

```
pool-4-thread-97号线程,长连接转换成短连接结果:keeuROXs  
pool-4-thread-482号线程,长连接转换成短连接结果:yGdfK0wD  
pool-4-thread-451号线程,长连接转换成短连接结果:hy3ZngD4  
1w并发存入长连接的集合大小: 10000  
1w并发存入长连接花费时间: 376
```

当然这个跟机器的性能有关系，但是综合发现，万级的并发读写能力都可以，在1s内能处理完成。

2.对map存储能力也做了测试（基于windows 8核16G测试）

发现map能存储的数据在百万级，当map存储个数超过了百万级别是，性能有明显下降，cpu飙升，而且内存急剧上升。

