短链接设计文档

一、解体思路

- 根据长链接的请求通过**雪花算法**给其分配一个随机的不重复且保证顺序的uuid，然后进行短链拼接；
- 通过继承**linkedHashMap**进行长短链接的，同时实现了**LRU算法**，尽可能防止JVM内存溢出。
- 在短链进行请求的时，先通过**布隆过滤器**将不存在的短链请求过滤掉；同时防止高并发导致服务不可用，使用了**令牌桶算法**进行限流。

1、链接转化思考

短链服务的核心就是构建短链接和长链接的唯一映射关系，生成这个长对短一一映射的关系中（可一长对多短），怎么能保证生成的短链接要是唯一的，并且应该如何存储在JVM内存中同时要防止内存溢出的问题。

- 1、转换的链地址要做到唯一，不能存在冲突。
  - 几种方法哈希计算、数据库自增主键、雪花算法、redis发号提升步长。综合考虑题意使用JVM存储一一映射关系，数据库和redis的方案就暂且不用。剩下hash计算，hash计算会出现

hash碰撞的问题。所以决定使用雪花算法保证生成唯一的id，然后通过62进制转换生成对应短链。

- 2、怎么存储到JVM中?
  - 这里只是借用于JVM内存，内存有限，并且要解决内存溢出的问题。
    - 短链和长链维护一一映射关系，自然想到Map，确定了存储结构。
    - 内存溢出的问题？内存本来就有限，肯定要对于内存数据进行有效期控制或者内存满了要有套淘汰策略。这里可以参考类似redis中缓存淘汰策略和过期key处理，为了简便的实现域名转换功能，决定可以参考使用**LRU算法**，内存将要打满的时候将最近最久没有使用的数据直接淘汰掉，另外也可以对于Map中的key设置一个有效的过期时间，但是数据多的话遍历的时间复杂度也提升了。
- 3、链接转换怎么保证接口稳定且高效?
  - 高效：链接获取的时候要将一些无用的请求链接过滤掉。长链接中不符合格式标准的，同时根据短链接获取场链接的时候也要将无用的短链不存在的短链过滤掉。一部分通过检测是否有效的链接，一部以过滤器以空间换时间提升效率。
  - 稳定：QPS要达到多少，做限流兜底，几种方案Hystrix、sentinal、还是**RateLimit**令牌桶算方法等。当前作业主要参考单体场景，所以就不做过重的设计，使用令牌桶做限流。
- 4、短链转长链的问题
  - 从JVM内存中拿到对应长链，是否需要额外进行重定向处理。考虑到作业中根据短链直接返回原始地址，开发流程上前端做处理可能更合适一点。

2、非功能性功能思考

限流：短链服务的主访问入口一般QPS极高，对其限流降低处理。

几种方案：1、sentinal限流 2、hystrix熔断降级，可设兜底方案 3、guava的令牌桶算法限流

轻量级的解决方案直接使用令牌桶算法进行限流。

性能：想尽一切办法降低该接口的耗时和提升转换效率。

1、使用多线程并发去做对链接转化，压榨服务器性能。

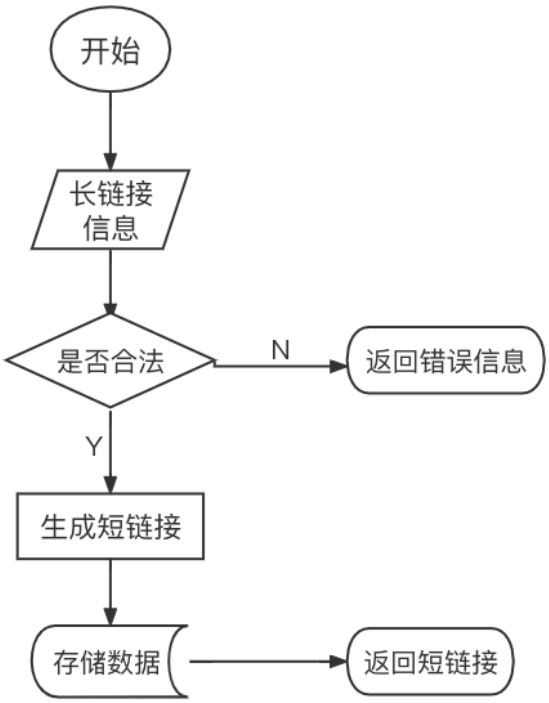
2、加层缓存。将无用请求直接过来掉，降低无用的接口耗时

3、稳定性。限流处理

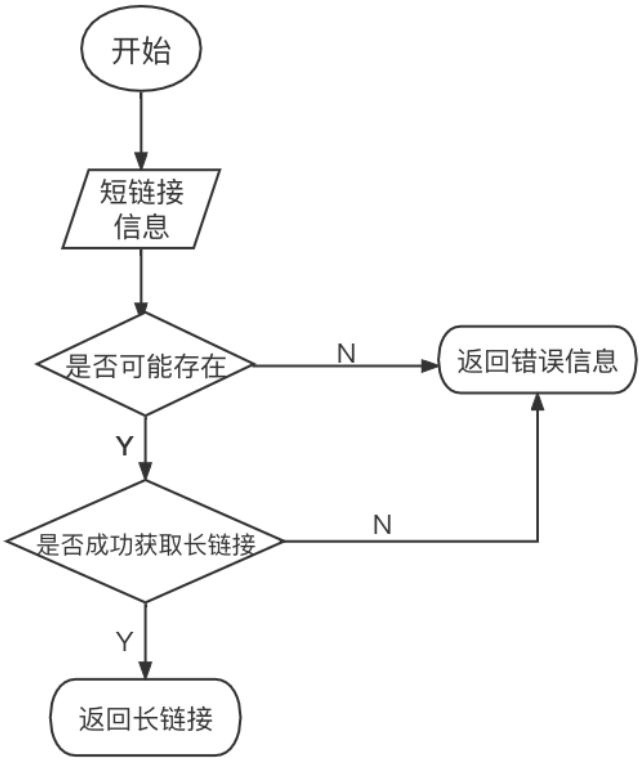
二、具体实现

1、流程图

服务的流程如下



短链接存储接口



短链接读取接口

2、测试

2.1 单元测试结果

Jacoco单元测试用例执行结果如下：其中分支覆盖率**90%**，行覆盖率接近**100%**。

hongshan

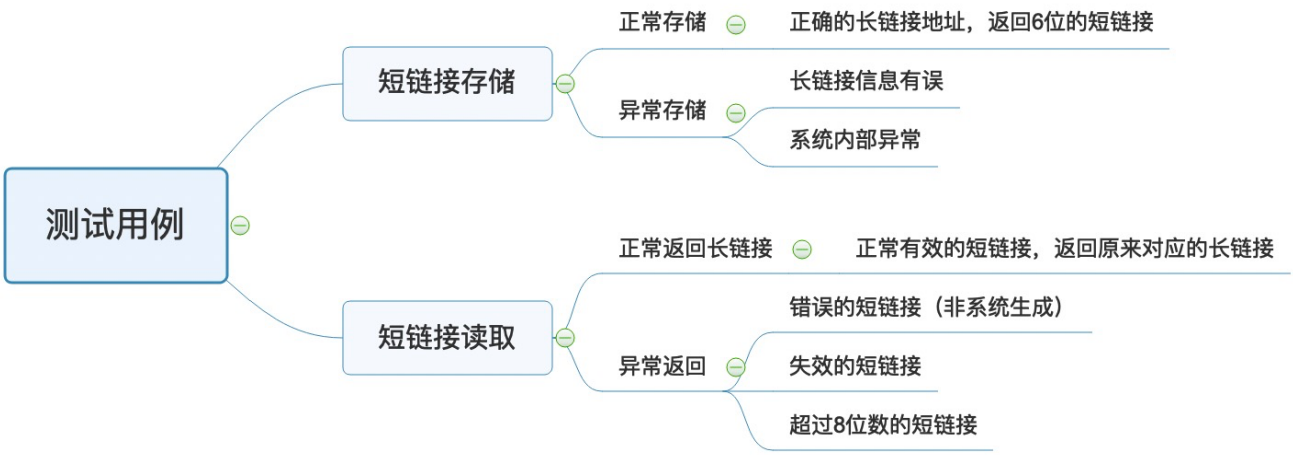
hongshan

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.yuanyang.hongshan.util		97%		100%	2	22	0	53	2	19	0	5
com.yuanyang.hongshan.service.impl		100%		100%	0	10	0	31	0	7	0	1
com.yuanyang.hongshan.controller		100%		75%	2	7	0	9	0	3	0	1
Total	6 of 430	98%	2 of 20	90%	4	39	0	93	2	29	0	7

idea集成mvn test，执行单元测试用例共计19个。

```
[INFO] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.014 s - in com.yuanyang.hongshan.UrlServiceImplTest
2021-12-23 16:19:19.803  INFO 94301 --- [extShutdownHook] o.s.s.concurrent.ThreadPoolTaskExecutor  : Shutting down ExecutorSe
icationTaskExecutor'
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 19, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- jacoco-maven-plugin:0.8.7:report (post-test) @ hongshan ---
[INFO] Loading execution data file /Users/yuanyang/Desktop/hongshan/target/jacoco.exec
[INFO] Analyzed bundle 'hongshan' with 7 classes
```

2.2 测试用例



测试案例

- 短链接读写接口正常返回

POST /url/generateShortURL generateShortURL

Response Class (Status 200)

OK

Model | Example Value

```
{
  "msg": "string",
  "rc": "ResultCode{code='0000', msg='成功'} SUCCESS",
  "rcAndMsg": "string",
  "success": true,
  "value": "string"
}
```

Response Content Type application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
requestDTO	{ "domain": "string", "longUrl": "https://www.sequoiacap.cn/china/people/introduction", "shortUrl": "string" }	requestDTO	body	Model | Example Value

```
{
  "domain": "string",
  "longUrl": "string",
  "shortUrl": "string"
}
```

Parameter content type: application/json

Response Messages

HTTP Status Code	Reason	Response Model	Headers
201	Created		
401	Unauthorized		
403	Forbidden		
404	Not Found		

Try it out! Hide Response

Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{ \
   "domain": "string", \
   "longUrl": "https://www.sequoiacap.cn/china/people/introduction", \
   "shortUrl": "string" \
 }' 'http://localhost:6699/url/generateShortURL'
```

Request URL

```
http://localhost:6699/url/generateShortURL
```

Request Headers

```
{
  "Accept": "application/json"
}
```

Response Body

```
{
  "value": "N01yq6",
  "rc": "SUCCESS",
  "msg": null,
  "success": true,
  "rcAndMsg": "ResultCode{code='0000', msg='成功'} SUCCESS-null"
}
```

Response Code

POST /url/getLongUrl getLongUrl

Response Class (Status 200)

OK

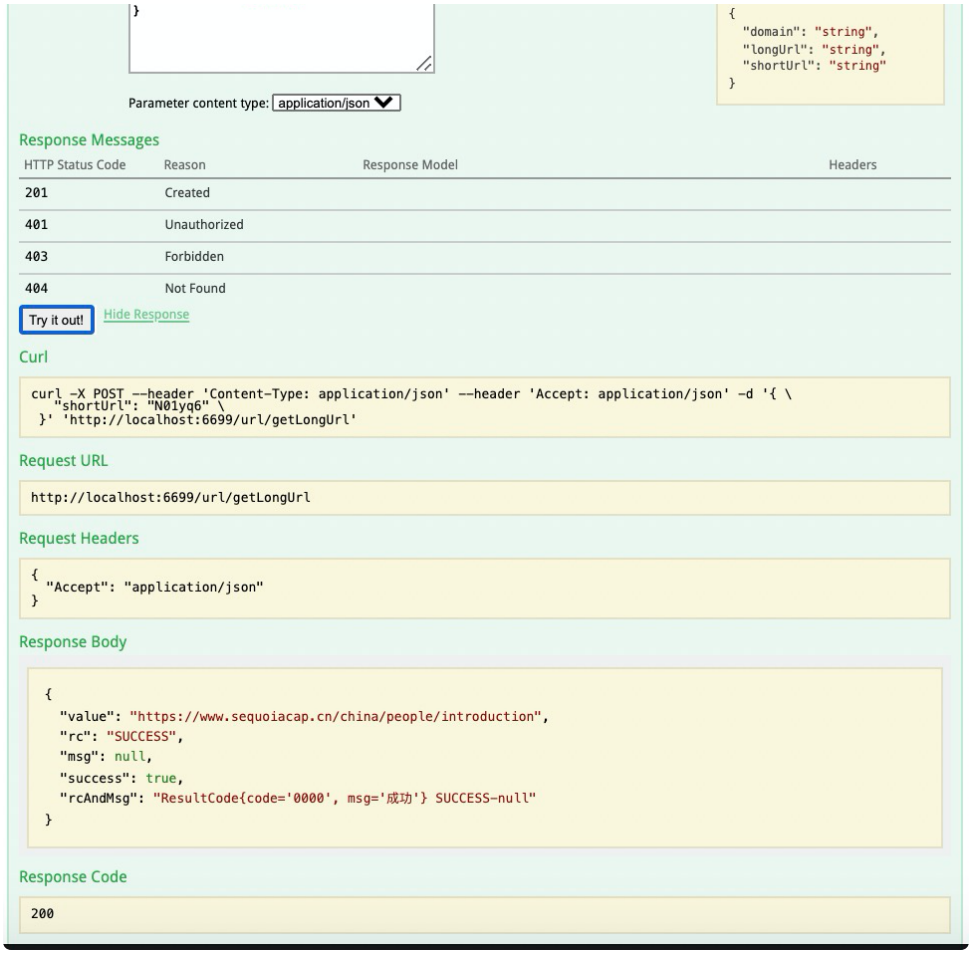
Model | Example Value

```
{
  "msg": "string",
  "rc": "ResultCode{code='0000', msg='成功'} SUCCESS",
  "rcAndMsg": "string",
  "success": true,
  "value": "string"
}
```

Response Content Type application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
requestDTO	{ "shortUrl": "N01yq6"	requestDTO	body	Model | Example Value


}
{
"domain": "string",
"longUrl": "string",
"shortUrl": "string"
}
Parameter content type: application/json
Response Messages
HTTP Status Code Reason Response Model Headers
201 Created
401 Unauthorized
403 Forbidden
404 Not Found
Try it out!
Hide Response
Curl
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{ \
"shortUrl": "N01yq6" \
}' 'http://localhost:6699/url/getLongUrl'
Request URL
http://localhost:6699/url/getLongUrl
Request Headers
{
"Accept": "application/json"
}
Response Body
{
"value": "https://www.sequoiacap.cn/china/people/introduction",
"rc": "SUCCESS",
"msg": null,
"success": true,
"rcAndMsg": "ResultCode{code='0000', msg='成功'} SUCCESS-null"
}
Response Code
200

异常结果

POST /url/getLongUrl getLongUrl

Response Class (Status 200)

OK

Model | Example Value

```
{
  "msg": "string",
  "rc": "ResultCode{code='0000', msg='成功'} SUCCESS",
  "rcAndMsg": "string",
  "success": true,
  "value": "string"
}
```

Response Content Type application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
requestDTO	{ "shortUrl": "N01yq611" }	requestDTO	body	Model | Example Value

```
{
  "domain": "string",
  "longUrl": "string",
  "shortUrl": "string"
}
```

Parameter content type: application/json

Response Messages

HTTP Status Code	Reason	Response Model	Headers
201	Created		
401	Unauthorized		
403	Forbidden		
404	Not Found		

Try it out! Hide Response

Curl

```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{ \
  "shortUrl": "N01yq611" \
}' 'http://localhost:6699/url/getLongUrl'
```

Request URL

```
http://localhost:6699/url/getLongUrl
```

Request Headers

```
{
  "Accept": "application/json"
}
```

Response Body

```
{
  "value": null,
  "rc": "ILLEGAL_DATA",
  "msg": null,
  "success": false,
  "rcAndMsg": "ResultCode{code='0003', msg='数据不存在或异常，请检查'} ILLEGAL_DATA-null"
}
```

Response Code

```
200
```

Response Class (Status 200)

OK

Model | Example Value

```
{
  "msg": "string",
  "rc": "ResultCode{code='0000', msg='成功'} SUCCESS",
  "rcAndMsg": "string",
  "success": true,
  "value": "string"
}
```

Response Content Type application/json

Parameters

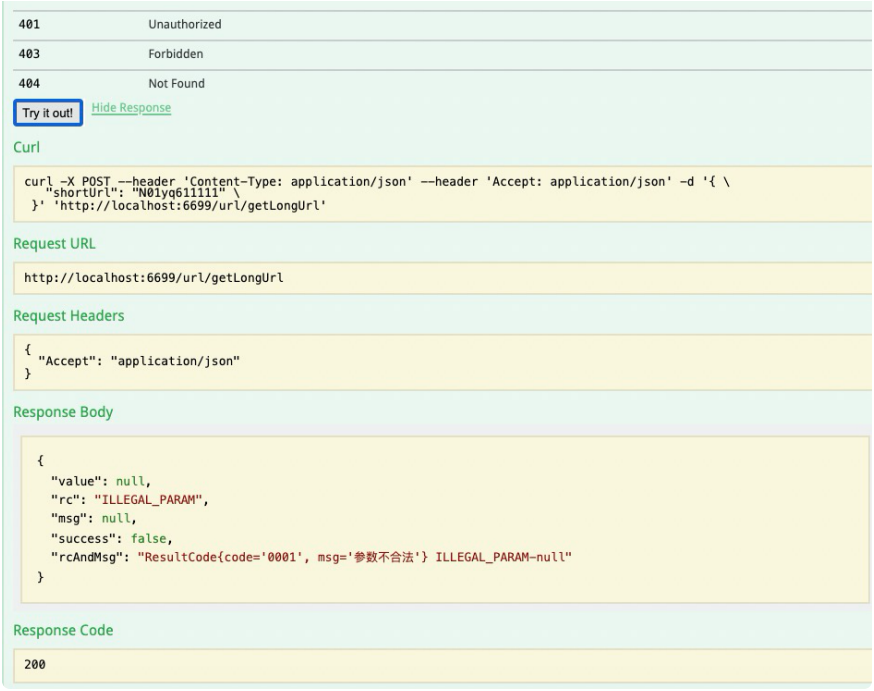
Parameter	Value	Description	Parameter Type	Data Type
requestDTO	{ "shortUrl": "N01yq611111" }	requestDTO	body	Model | Example Value

```
{
  "domain": "string",
  "longUrl": "string",
  "shortUrl": "string"
}
```

Parameter content type: application/json

Response Messages

HTTP Status Code	Reason	Response Model	Headers
201	Created		



2.3 性能测试

tps最高达到**2000**左右，随并发梯度增加很稳定。

具体见性能测试报告。

三、扩展设计

1、设计优点

在满足题意的情况下，短链服务中的一些场景问题基本都已经覆盖到了，比较轻量级的实现方式，在短链接服务正常使用的的同时尽可能保证服务安全。

1. 在基于JVM内存的情况下，继承现有的LinkedHashMap去实现LRU算法去存储短链接和长链接的映射关系，解决内存溢出问题；
2. 提升服务整体的吞吐量，对无效的链接进行过滤，同时使用了布隆过滤器针过滤掉无用的短链接请求，提高效率
3. 使用雪花算法生成唯一uuid，转换62进制，保证每个短链的唯一性。即使后期升级为集群环境部署，雪花算法依旧有效
4. 保证服务稳定，针对大的QPS场景为了保证服务稳定性，使用了限流机制，令牌桶算法。

2、设计缺点及扩展

缺点

a. 对短链服务其实还有很多工作。例如本次homework中的非功能性需求，例如拦截器的思想，这里主要提供了一个校验的判断，作为一道简单的拦截，对域名和ip地址基本没有做什么校验。可能某个时间同一个ip疯狂请求接口导致服务不可用。

b. 没有兜底方案的设计，如果在某段时间中大量请求失败接口暂时服务，采用兜底方案，直到某阶段接口返回正确率达到50%以上在重新恢复正常逻辑。虽然现在有做限流，但是主要针对单机情况，集群情况下需要对方案进行升级，集群环境下进一步限流。

优化扩展：

a. 真正在后期项目中，拦截器的工作其实还可以做很多，基本容器的拦截器负责服务安全，例如黑白名单域名、黑白明白URI等；另外还有一些服务内部自定义的拦截器链，实现请求参数解析、URL转换等拦截功能。提升系统的稳定性和安全性的时候降低无用的请求提升性能；同时可在nginx针对单ip设置访问流量，超过阈值可以有一些防护策略，例如生成短链接数据接口要限制在10秒中只能生成短链接一次；

b. 目前是基于LRU算法做的缓存，其实这块市场上有很多种方式。本次HomeWork使用的JVM缓存，对于API的使用直接继承了LinkedHashMap去完成，这里是必然有局限性的，所以真实项目中会优先考虑使用Redis进行Map<短链接，长链接>的存储，这种场景使用Redis做缓存非常贴合，一部分可以根据业务的情况有效的控制短链的有效时间，另外Redis本身就自带内存淘汰策略(LRU其中一种)，同时对于过期的key本身也会有过期策略清理的方式。但是有个问题，Redis会出现缓存穿透和雪崩的问题，虽然布隆过滤器可解决穿透问题，雪崩的情况对应key的过期时间要尽可能打散，用集群的模式也可解决。

c. 在真正的大并发的情况下，是否要考虑的加锁的问题。那么锁什么？

高并发的情况下确实是不安全的，那么是否有必要引入分布式锁或者本地锁。目前看来限流了之后其实并没有多线程竞争共享资源的情况，所以不需要额外考虑。如果哪天要把限流完全放开，那么这个地方就需要引入锁了，自定义一个key，通过获取这个key在加锁和设置有效时间。

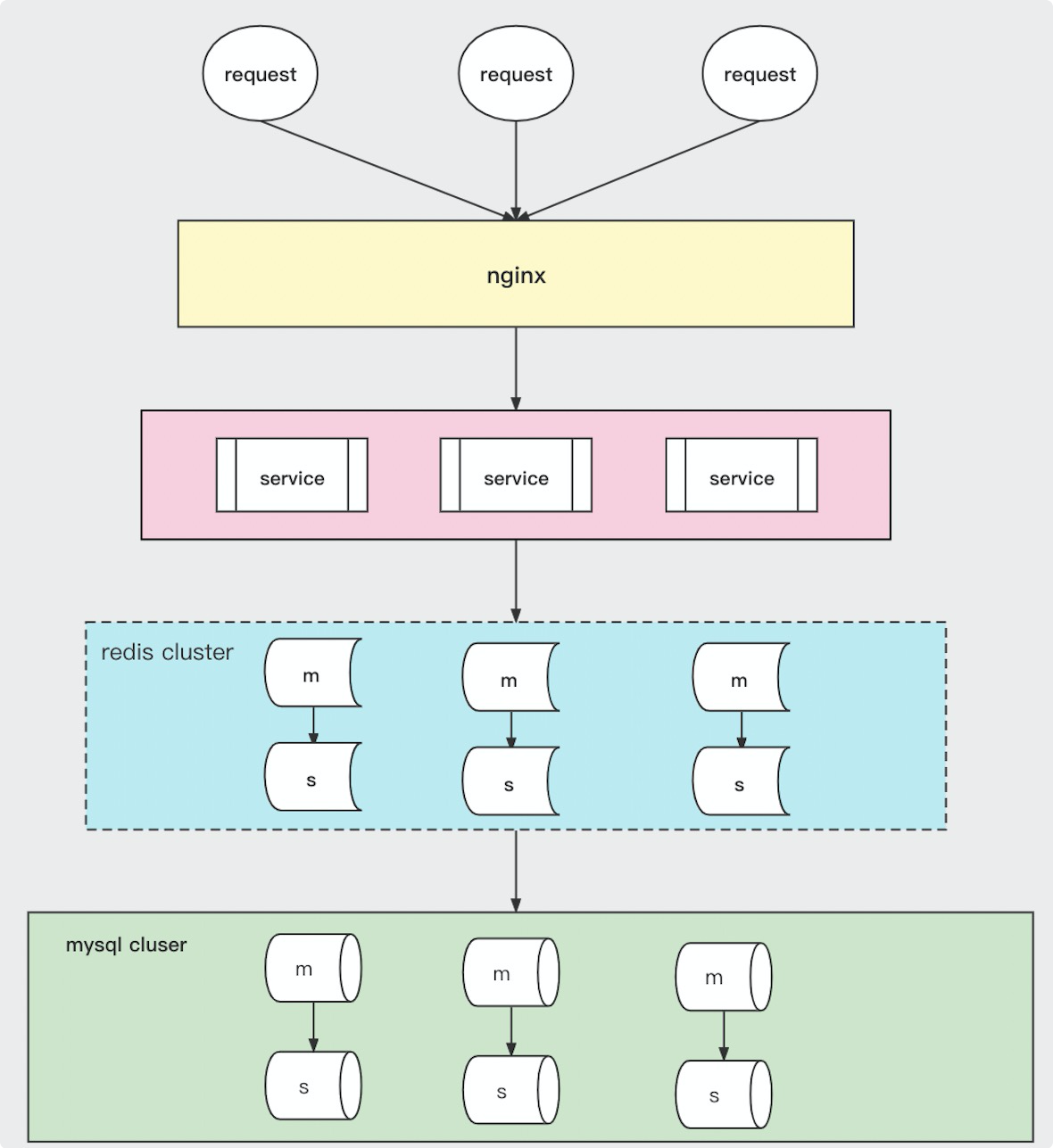
d. 本地只是基于单机用了限流的处理。集群模式下需要进行新的设计方案，可引用nginx、sential、hystrix等限流降低机制，保证服务稳定性；做到服务的高可用，集群环境的搭建以及在流量入口去实现负载均衡。

3、后续架构选型思考

结合自己的一些思考，在保证短链接服务的高可用性和稳定性，另外对数据要进行持久化。

1. **nginx**做流量分发和负载均衡以及限流。（暂**不使用sential等中间件避免过度设计**）
2. 过滤器，对于无效请求包括一些限制的黑白名单等过滤器处理，和实际业务的过滤器
3. redis集群搭建，缓存短链和长链的关系。本服务**不需要解决数据不一致性**问题，高并发下需要考虑缓存穿透问题（**布隆过滤器**解决），至于**缓存击穿**的情况大概率不会出现（如果业务需要加分布式锁）。
4. mysql主从持久化。

架构设计图


request
request
request
nginx
service
service
service
redis cluster
m
m
m
s
s
s
mysql cluser
m
m
m
s
s
s