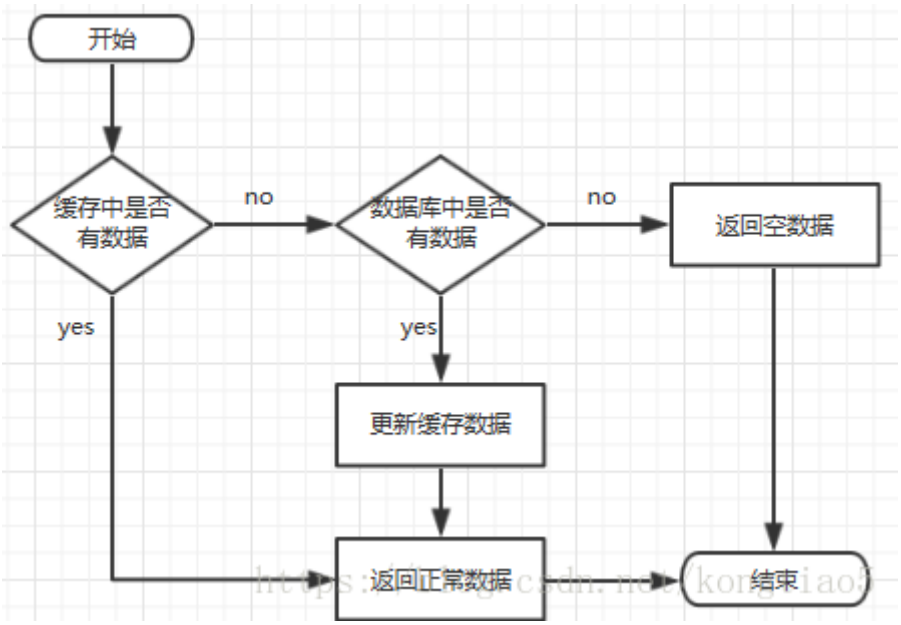
Redis缓存穿透、击穿、雪崩

缓存系统示意图：



缓存穿透：

缓存穿透是指缓存和数据库中都没有的数据，而用户不断发起请求。导致每次访问都是直接访问存储层的数据库。

解决方案：

1）接口层添加校验：给id添加基本校验，id<=0的直接拦截

2）对查询结果为空的情况也进行缓存（设置为key-null），缓存时间设置短一点。

3）使用布隆过滤器，快速判断一个key是否在容器中，不存在就直接返回。

缓存击穿：

缓存击穿是指缓存中没有但数据库中有的数据（一般是缓存时间到期），这时由于并发用户特别多，同时读缓存没读到数据，又同时去数据库去取数据，引起数据库压力瞬间增大，造成过大压力。

解决方案：

1）设置热点数据永不过期（Expire key time）

缓存雪崩：

缓存雪崩是指缓存中数据大批量到过期时间，而查询数据量巨大，引起数据库压力过大甚至down机。和缓存击穿不同的是，缓存击穿指并发查同一条数据，缓存雪崩是不同数据都过期了，很多数据都查不到从而查数据库。

解决方案：

1）缓存数据的过期时间设置地尽量分散一点，防止大批量缓存同时失效。比如我们可以在原有的失效时间基础上增加一个随机值。

redis脑裂：

redis的集群脑裂是指因为网络问题，导致redis master节点跟redis slave节点和sentinel集群处于不同的网络分区，此时因为sentinel集群无法感知到master的存在，所以将slave节点提升为master节点。

此时同时存在2个master节点，就像一个大脑分裂成了两个。

集群脑裂问题中，如果客户端还在基于原来的master节点继续写入数据，那么新的master节点将无法同步这些数据，当网络问题解决之后，sentinel集群将原先的master节点降为slave节点，此时再从新的master中同步数据，将会造成大量的数据丢失。

解决方案：

redis的配置文件中，存在两个参数

```
min-slaves-to-write 3
min-slaves-max-lag 10
```

第一个参数表示连接到master的最少slave数量

第二个参数表示slave连接到master的最大延迟时间

如果连接到master的slave数量小于第一个参数，且ping的延迟时间小于等于第二个参数，那么master就会拒绝写请求，配置了这两个参数之后，如果发生集群脑裂，原先的master节点接收到客户端的写入请求会拒绝，就可以减少数据同步之后的数据丢失。

注意：较新版本的redis.conf文件中的参数变成了

```
min-replicas-to-write 3
min-replicas-max-lag 10
```

redis中的异步复制情况下的数据丢失问题也能使用这两个参数

redis常用命令：

启动redis：# redis-server

如何保持数据库与redis中数据的一致性？

(https://www.cnblogs.com/AshOfTime/p/10815593.html)

1）每次写入数据库成功，即让缓存失效，下次读取时在缓存，这是缓存的实时策略（先写数据库，在删缓存 Cache Aside Pattern）（问题：更新时序问题　设置redis数据过期时

间）

2）先删缓存，在更新数据库（问题：时序混乱读取到脏数据 解决方案：延迟双删）

1.先删除缓存。

2.写入数据库

3.休眠一秒。执行删除缓存(目的是把1秒内产生的脏数据重新从缓存中删除)

3）Write Behind Caching Pattern--只更缓存，不更MySQL，MySQL由缓存异步的更新

4）Redis里的数据不立刻更新，等redis里数据自然过期。然后去DB里取，顺带重新set redis（会有一段时间出现数据不一致问题）

Redis为什么这么快

1、完全基于内存，绝大部分请求是纯粹的内存操作，非常快速。数据存在内存中，类似于HashMap，HashMap的优势就是查找和操作的时间复杂度都是O(1)；

2、数据结构简单，对数据操作也简单，Redis中的数据结构是专门进行设计的；

3、采用单线程，避免了不必要的上下文切换和竞争条件，也不存在多进程或者多线程导致的切换而消耗 CPU，不用去考虑各种锁的问题，不存在加锁释放锁操作，没有因为可能出现死锁而导致的性能消耗；

4、使用多路I/O复用模型，非阻塞IO；

5、使用底层模型不同，它们之间底层实现方式以及与客户端之间通信的应用协议不一样，Redis直接自己构建了VM 机制 ，因为一般的系统调用系统函数的话，会浪费一定的时间去移动和请求；