# 分布式锁

1. 本地缓存与分布式缓存
   1. 缓存适用场景
      1. 即时性，数据一致性要求不高
      2. 访问量大且更新频率不高的
   2. 流程：发送请求-》查看缓存中是否有数据
      1. 有，直接返回
      2. 没有，查询数据库-》将数据放入缓存（设置过期时间）-》返回
   3. 本地缓存基于内存（Map）
      1. 分布式下会对一个服务进行集群，当负载均衡时，本地缓存就失效了
      2. 更严重的是若修改了数据，一般会将缓存中也修改。如果是本地缓存就会出现在负载均衡情况下的数据不一致性
2. bug
   1. 压测出的内存泄露及解决
      1. 用Jmter测出堆外内存溢出
      2. 先查看一下自己jvm参数是否设小了，设大一点
      3. 再次通过jmter进行大并发测试，只是延缓了出现异常的时间，还是出现了这个问题
      4. 原因是boot2.0之后默认使用Lettuce为操作redis的客户端
      5. Lettuce并没有直接释放连接，可以切换为jedis
3. 雪崩，击穿，穿透
   1. 穿透
      1. 查一个不存在的数据，因为缓存中也没有，所以就得去数据库查
      2. 但是数据库也没有啊，所以就导致每次请求的要去DB查
      3. 万一被坏人利用这个不存在的数据进行恶意攻击，那么DB压力瞬间就增大，导致崩溃
      4. 解决：可以在查数据库的时候，给不存在的数据加个标志位（设个默认值），但是要加个过期时间噢，万一数据库有了也可以同步上来
   2. 雪崩
      1. 在同一时刻，缓存同时过期，请求就全部到了DB，DB瞬间被压垮
      2. 解决：可以在设置过期时间的时候，加个随机值呀
   3. 击穿
      1. 例如噢，苹果13刚刚发布，它热度特别高，就是个热点数据了，每天都很多人关注
      2. 它也有个过期时间不，一旦过期，突然大量并发冲进来，查缓存并没有
      3. 于是又去了数据库，又崩溃了
      4. 解决：加锁，我可以加上一把锁，来了一百万请求，我只让第一个进去，然后查询完之后把结果放入缓存，然后释放锁，下一个拿到锁就去缓存看看有没有数据，这就做到了防止击穿
4. 本地锁在分布式环境下的问题
   1. 其实也能锁，但是不能完全锁住
   2. 比如我现在有四台集群服务，来了一百万请求，我是本地锁嘛，相当于有四把锁，我得放四个过去，要想完全锁住还得是分布式锁
   3. 时序问题：就在 确认缓存中有没有数据，查DB，存进缓存，他这个过程得是原子性的呀，不然在请求redis时可能会网络不好，然后下一个请求直接进来了
5. 分布式锁
   1. 就像是给众多服务提供了一个公共厕所
   2. 所有的服务都去一个地方上厕所，去了一个就把门锁上，执行逻辑，如果没有抢到，那就只能等上一位拉完释放锁才能继续了
   3. 这个厕所可以由redis来提供，setnx（“lock”，111）在Java中的方法是setIfabsent（）
   4. 第一种使用方法
      1. 首先调用redis，设置一个锁，setnx如果说锁上了就会返回ok，否则返回的是null
      2. 返回ok，则证明获取到锁了，就可以执行下一步逻辑，执行完记得删除锁，走出厕所，给兄弟们条活路
      3. 返回null，则证明获取失败，那么就继续等待呗（可以使用自旋的方式，等待几秒钟，自己再调一波自己）
      4. 问题1
         1. 如果执行业务逻辑的时候发生宕机就会导致锁没删除，变成了死锁
         2. 解决方案：可以加一个过期时间，但是如果在设置锁占位的时候发生宕机，那么还是死锁，所有这个占位和设置过期时间必须是一个原子性的
      5. 问题2
         1. 解决了加锁时原子性，那么在删锁的时候也会出现问题
         2. 就是说业务执行时间比设置的过期时间要长，业务还没执行完，锁就没了，那么下一个请求一看，哎，终于到我了，于是它抢到了，当业务刚刚好执行完，跨擦一删，删掉的是别人的锁
         3. 问题：要加锁时加个UUID，要删锁时，从redis中取出来做一次比较，但是这个取的过程可能会出现这种情况
         4. 正在判断当前值，要删除时，锁就刚刚好过期了，新的值就设置进来了，则删除的将是别人的锁
         5. 解决：redis+lua脚本
   5. Redisson 可重入锁
      1. lock（）占位，unlock（）解锁
      2. 不会发生死锁问题，它有一个看门狗机制，会自动续期
      3. 看门狗机制
         1. 如果传递了超时时间，那么就不会自动续期了
         2. 如果未指定，它就有一个变量值是30 \* 1000（30s）的默认时间
         3. 只要占锁成功，就会启动一个定时任务，每隔10秒就会自动调用一次
   6. 缓存一致性问题
      1. 双写模式
         1. 修改完数据之后，也需要将缓存中也修改同步一下
      2. 失效模式
         1. 修改完数据之后，直接删掉
      3. 解决方案
         1. 加上过期时间即可，每隔一段时间就会触发读的自动更新
         2. 通过加 读(相当于无锁状态)写锁
         3. 阿里开源的canal
         4. 遇到一致性要求高的就应该查数据库，慢一点也没关系