**场景1）**

**解决方案：**

1. 设计计数器，记录调用次数，用于控制业务执行次数。以用户id作为key，使用次数作为value
2. 在调用前获取次数，判断是否超过限定次数

不超过次数的情况下，每次调用计数+1

超过，业务调用失败，计数-1

1. 为计数器设置生命周期为指定周期，如：1min等。自动清空周期内使用次数

**方案改良：**

**取消最大值的判定，利用incr操作超过最大值抛异常的形式替代每次判断**

**判断是否为nil**

**是 -- 设置为Max - 次数**

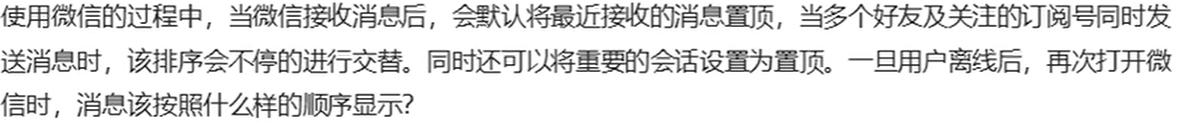
**否 -- 计数 + 1**

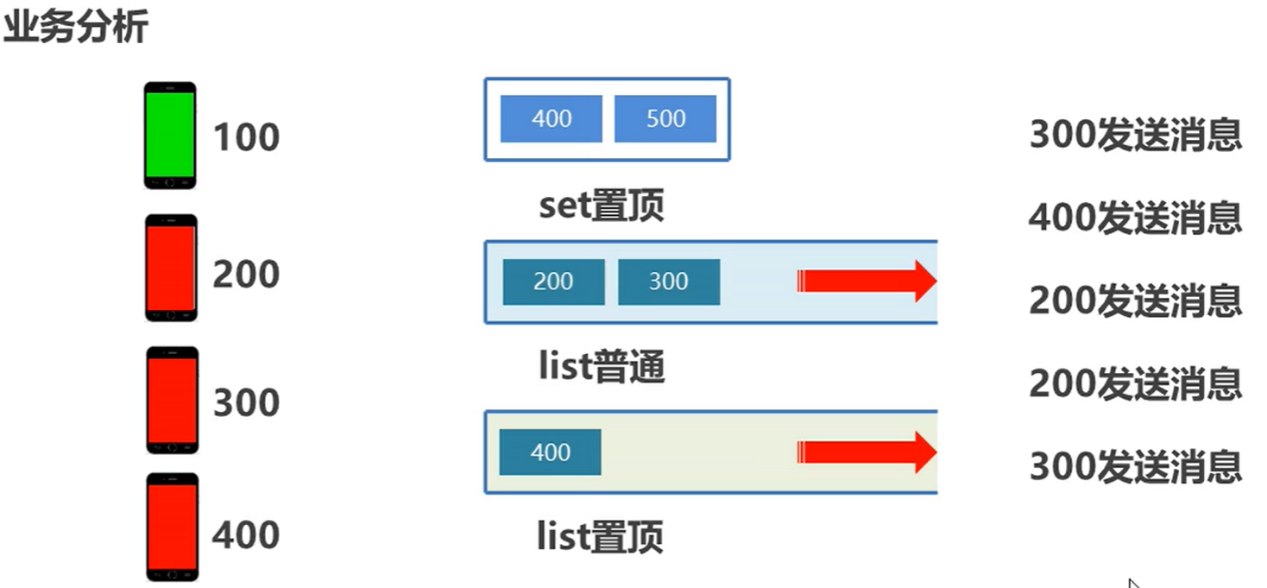
**业务调用失败，计数 - 1**

**遇到异常即 + 1操作超过上限，视为使用达到上限**

**ps16：**

**redis应用于显示按次计算的服务控制**

**场景2）**



描述：100 是自己的手机，其他都是好友的手机，模拟好友给自己发消息

set保存置顶的人员编号

list普通 -- 保存普通人员消息队列

list置顶 -- 保存指定人员消息队列

当相同的编号发送消息的时候，先将其从消息队列中删除后再添加进去

**解决方案：**

1. 依赖list的数据具有顺序的特征对消息进行管理，将list结构作为栈使用
2. 对置顶与普通会话分别创建独立的list分别管理
3. 当某个list中接收到用户消息后，将消息发送方的id从list一侧加入list（lpush）
4. 多个相同id发出的消息反复入栈会出现问题，在入栈之前无论是否具有当前id对应的消息，先删除对应id
5. 推送消息时先推送置顶会话list，再推送普通会话list，推送完成的list清除所有数据
6. 消息的数量，也就是微信用户对话数量采用计数器的思想另行记录，伴随list操作同步更新

**ps17：**

**redis应用于基于时间顺序的数据操作，而不关注具体时间**

