限流的算法有哪些?

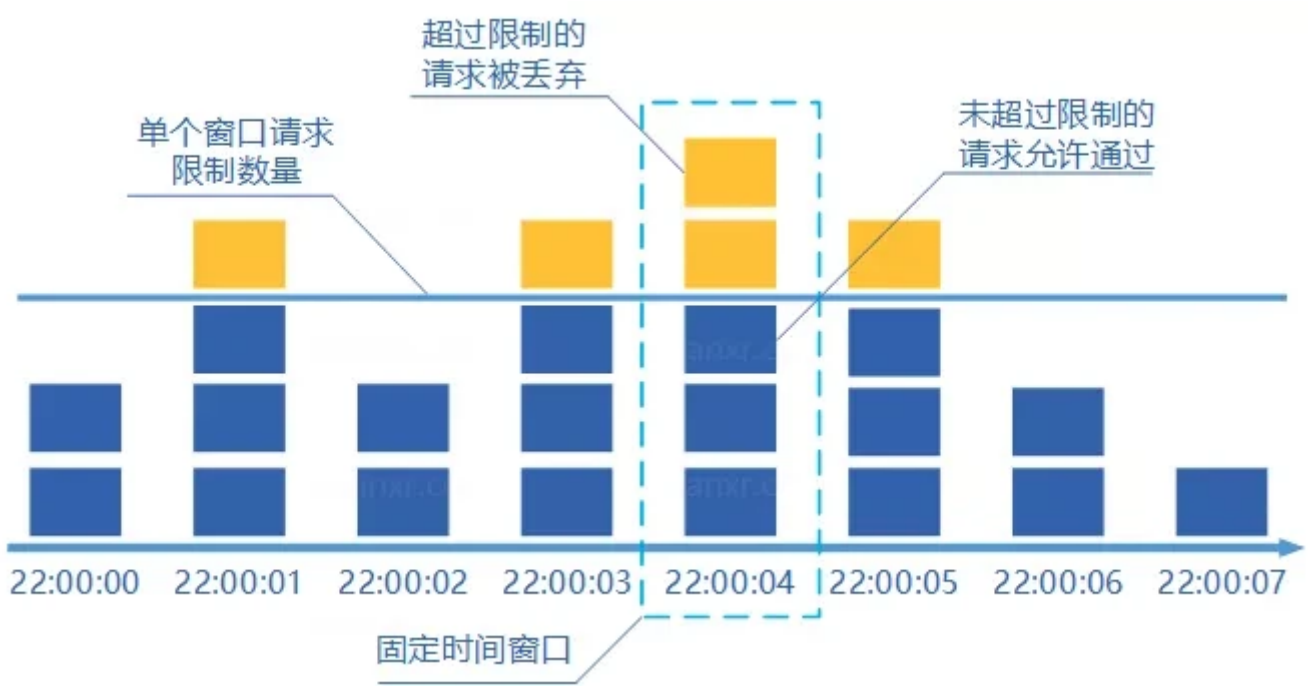
简单介绍 4 种非常好理解并且容易实现的限流算法!

下图的图片不是 Guide 哥自己画的哦!图片来源于 InfoQ 的一篇文章《分布式服务限流实战，已经为你排好坑了》。

固定窗口计数器算法

该算法规定我们单位时间处理的请求数量。比如我们规定我们的一个接口一分钟只能访问10次的话。使用固定窗口计数器算法的话可以这样实现：给定一个变量counter来记录处理的请求数量，当1分钟之内处理一个请求之后counter+1，1分钟之内的如果counter=100的话，后续的请求就会被全部拒绝。等到 1分钟结束后，将counter回归成0，重新开始计数（ps：只要过了一个周期就讲counter回归成0）。

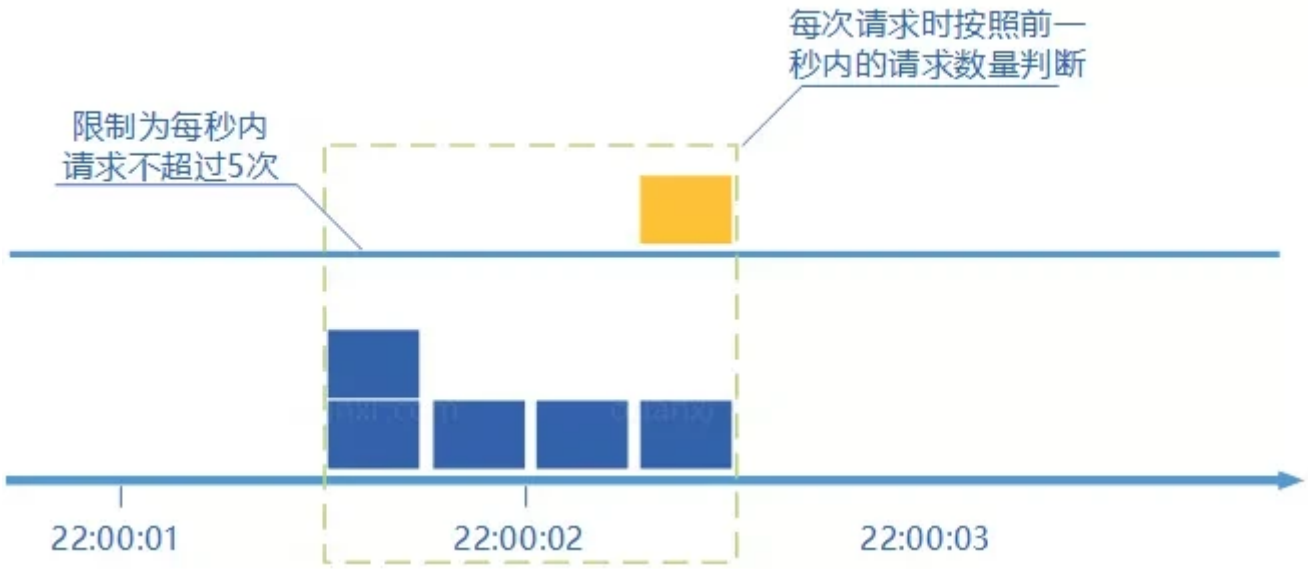
这种限流算法无法保证限流速率，因而无法保证突然激增的流量。比如我们限制一个接口一分钟只能访问10次的话，前半分钟一个请求没有接收，后半分钟接收了10个请求。



滑动窗口计数器算法

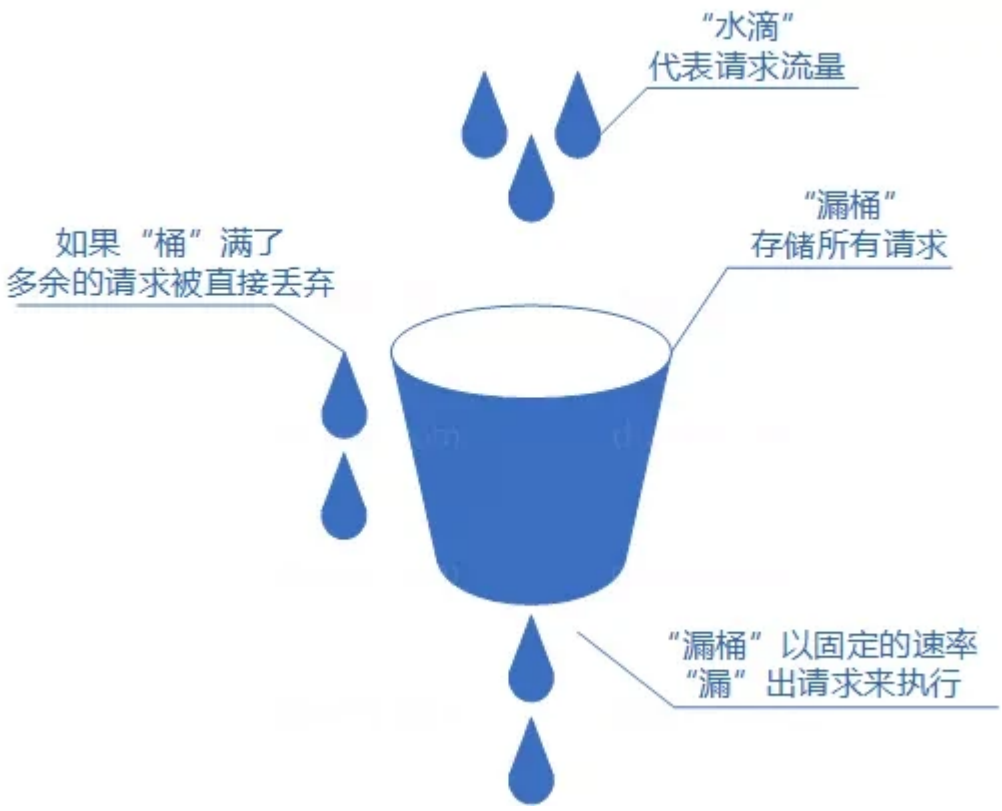
该算法算的上是固定窗口计数器算法的升级版。滑动窗口计数器算法相比于固定窗口计数器算法的优化在于：它把时间以一定比例分片。例如我们的接口限流每分钟处理60个请求，我们可以把 1 分钟分为60个窗口。每隔1秒移动一次，每个窗口一秒只能处理 不大于 60(请求数)/60（窗口数） 的请求， 如果当前窗口的请求计数总和超过了限制的数量的话就不再处理其他请求。

很显然：当滑动窗口的格子划分的越多，滑动窗口的滚动就越平滑，限流的统计就会越精确。



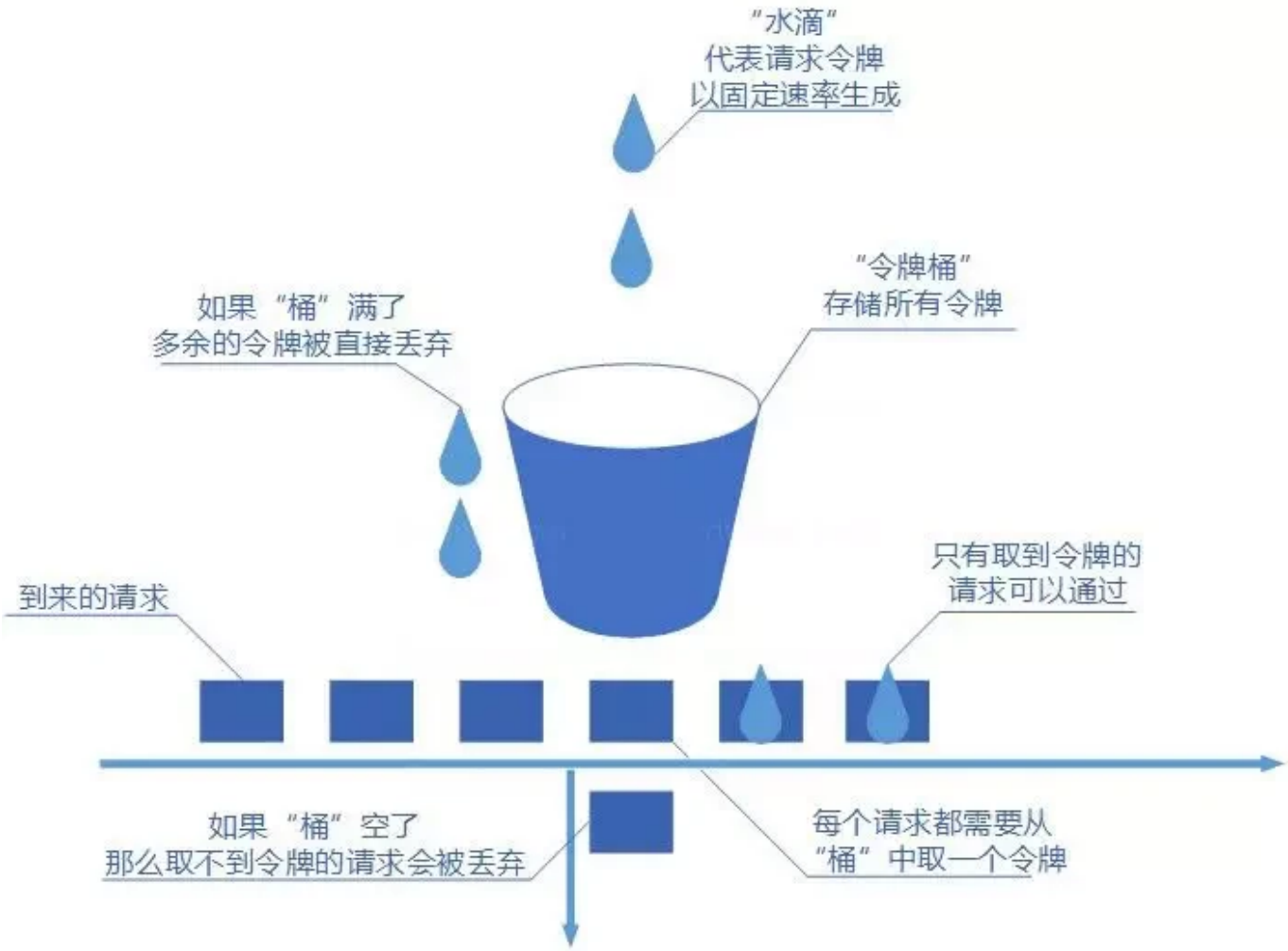
漏桶算法

我们可以把发请求的动作比作成注水到桶中，我们处理请求的过程可以比喻为漏桶漏水。我们往桶中以任意速率流入水，以一定速率流出水。当水超过桶流量则丢弃，因为桶容量是不变的，保证了整体的速率。如果想要实现这个算法的话也很简单，准备一个队列用来保存请求，然后我们定期从队列中拿请求来执行就好了。



令牌桶算法

令牌桶算法也比较简单。和漏桶算法算法一样，我们的主角还是桶（这限流算法和桶过不去啊）。不过现在桶里装的是令牌了，请求在被处理之前需要拿到一个令牌，请求处理完毕之后将这个令牌丢弃（删除）。我们根据限流大小，按照一定的速率往桶里添加令牌。


"水滴"
代表请求令牌
以固定速率生成
"令牌桶"
存储所有令牌
如果"桶"满了
多余的令牌被直接丢弃
只有取到令牌的
请求可以通过
到来的请求
如果"桶"空了
那么取不到令牌的请求会被丢弃
每个请求都需要从
"桶"中取一个令牌