概念：过滤不合理的请求，保证系统流量在可控范围之内

限流方法：

1：漏桶：

2：令牌桶：RateLimiter rateLimiter = RateLimiter.*create*(10);

常用的限流算法有两种：漏桶算法和令牌桶算法。

      漏桶算法思路很简单，水（请求）先进入到漏桶里，漏桶以一定的速度出水，当水流入速度过大会直接溢出，可以看出漏桶算法能强行限制数据的传输速率。

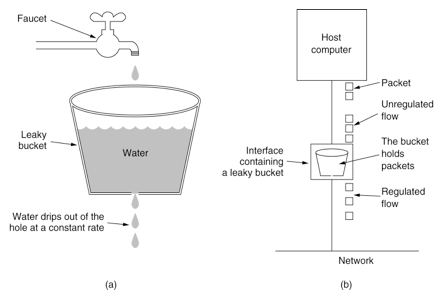


图1 漏桶算法示意图

      对于很多应用场景来说，除了要求能够限制数据的平均传输速率外，还要求允许某种程度的突发传输。这时候漏桶算法可能就不合适了，令牌桶算法更为适合。如图2所示，令牌桶算法的原理是系统会以一个恒定的速度往桶里放入令牌，而如果请求需要被处理，则需要先从桶里获取一个令牌，当桶里没有令牌可取时，则拒绝服务。

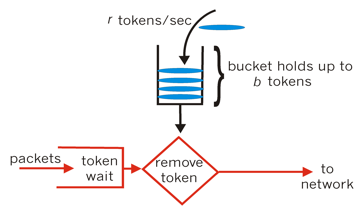


图2 令牌桶算法示意图

当网络设备衡量流量是否超过额定带宽时，需要查看令牌桶，而令牌桶中会放置一定数量的令牌，一个令牌允许接口发送

　　或接收1bit数据（有时是1 Byte数据），当接口通过1bit数据后，同时也要从桶中移除一个令牌。当桶里没有令牌的时候，任何流

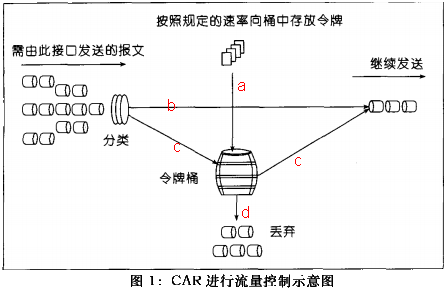
　　量都被视为超过额定带宽,只有当桶中有令牌时，数据才可以通过接口。令牌桶中的令牌不仅仅可以被移除，同样也可以往里添加，

　　所以为了保证接口随时有数据通过，就必须不停地往桶里加令牌，由此可见，往桶里加令牌的速度，就决定了数据通过接口的速度。

　　因此，我们通过控制往令牌桶里加令牌的速度从而控制用户流量的带宽。而设置的这个用户传输数据的速率被称为承诺信息速率（CIR），

　　通常以秒为单位。比如我们设置用户的带宽为1000  bit每秒，只要保证每秒钟往桶里添加1000个令牌即可。

令牌桶算法图例



　　　　a. 按特定的速率向令牌桶投放令牌

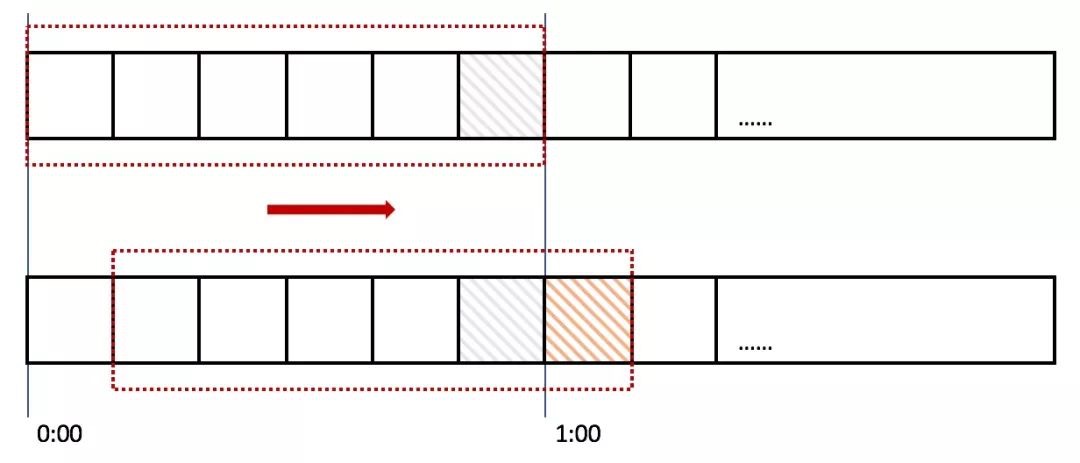
　　　　b. 根据预设的匹配规则先对报文进行分类，不符合匹配规则的报文不需要经过令牌桶的处理，直接发送；

　　　　c. 符合匹配规则的报文，则需要令牌桶进行处理。当桶中有足够的令牌则报文可以被继续发送下去，同时令牌桶中的令牌 量按报文的长度做相应的减少；

　　　　d. 当令牌桶中的令牌不足时，报文将不能被发送，只有等到桶中生成了新的令牌，报文才可以发送。这就可以限制报文的流量只能是小于等于令牌生成的速度，达到限制流量的目的。

**3、滑动窗口**

滑动窗口，又称rolling window。如果学过TCP网络协议，一定对滑动窗口这个名词不会陌生。下图很好地解释了滑动窗口算法



在上图中，整个红色的矩形框表示一个时间窗口，在这个例子中，一个时间窗口就是一分钟。我们将滑动窗口划成6格，每格代表10秒钟。每过10秒钟，时间窗口就会往右滑动一格。每一个格子都有自己独立的计数器counter，比如一个请求在0:35秒时到达，那么0:30~0:39对应的counter就会加1。

滑动窗口怎么解决刚才的临界问题的呢？我们可以看上图，0:59到达的100个请求会落在灰色的格子中，而1:00到达的请求会落在橘黄色的格子中。当时间到达1:00时，窗口会往右移动一格，此时时间窗口内的总请求数量一共是200个，超过了限定的100个，所以此时能够检测出来触发了限流。

回顾一下刚才的计数器算法，其实就是滑动窗口。只是它没有对时间窗口做进一步地划分，只有1格。

由此可见，滑动窗口格子划分越多，窗口滚动就越平滑，限流统计就会越精确。